[MS-WSP-Diff]:

Windows Search Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
12/18/2006	0.1	New	Version 0.1 release
3/2/2007	1.0	Major	Version 1.0 release
4/3/2007	1.1	Minor	Version 1.1 release
5/11/2007	1.2	Minor	Version 1.2 release
6/1/2007	1.2.1	Editorial	Changed language and formatting in the technical content.
7/3/2007	1.3	Minor	Minor technical content changes.
7/20/2007	1.3.1	Editorial	Changed language and formatting in the technical content.
8/10/2007	1.4	Minor	Clarified the meaning of the technical content.
9/28/2007	1.5	Minor	Revised technical and editorial content based on feedback.
10/23/2007	1.5.1	Editorial	Changed language and formatting in the technical content.
11/30/2007	2.0	Major	Revised technical content based on feedback.
1/25/2008	2.1	Minor	Clarified the meaning of the technical content.
3/14/2008	2.1.1	Editorial	Changed language and formatting in the technical content.
5/16/2008	2.1.2	Editorial	Changed language and formatting in the technical content.
6/20/2008	3.0	Major	Updated and revised the technical content.
7/25/2008	4.0	Major	Updated and revised the technical content.
8/29/2008	5.0	Major	Updated and revised the technical content.
10/24/2008	6.0	Major	Updated and revised the technical content.
12/5/2008	7.0	Major	Updated and revised the technical content.
1/16/2009	8.0	Major	Updated and revised the technical content.
2/27/2009	9.0	Major	Updated and revised the technical content.
4/10/2009	10.0	Major	Updated and revised the technical content.
5/22/2009	11.0	Major	Updated and revised the technical content.
7/2/2009	12.0	Major	Updated and revised the technical content.
8/14/2009	13.0	Major	Updated and revised the technical content.
9/25/2009	14.0	Major	Updated and revised the technical content.
11/6/2009	15.0	Major	Updated and revised the technical content.
12/18/2009	16.0	Major	Updated and revised the technical content.
1/29/2010	17.0	Major	Updated and revised the technical content.
3/12/2010	18.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
4/23/2010	19.0	Major	Updated and revised the technical content.
6/4/2010	20.0	Major	Updated and revised the technical content.
7/16/2010	21.0	Major	Updated and revised the technical content.
8/27/2010	21.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2010	21.0	None	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	21.1	Minor	Clarified the meaning of the technical content.
1/7/2011	22.0	Major	Updated and revised the technical content.
2/11/2011	23.0	Major	Updated and revised the technical content.
3/25/2011	23.0	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	24.0	Major	Updated and revised the technical content.
6/17/2011	25.0	Major	Updated and revised the technical content.
9/23/2011	25.0	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	26.0	Major	Updated and revised the technical content.
3/30/2012	26.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	26.1	Minor	Clarified the meaning of the technical content.
10/25/2012	26.1	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	26.1	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	27.0	Major	Updated and revised the technical content.
11/14/2013	27.0	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	27.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	27.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	28.0	Major	Significantly changed the technical content.
10/16/2015	29.0	Major	Significantly changed the technical content.
7/14/2016	30.0	Major	Significantly changed the technical content.
6/1/2017	30.0	None	No changes to the meaning, language, or formatting of the technical content.
9/15/2017	31.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

This document specifies the Windows Search Protocol, which allows a client to issue queries to a server hosting a Generic Search service (GSS). The protocol is primarily intended to be used for full-text queries. It also allows an administrator to remotely manage the GSS.

This document specifies both remote querying and remote administration of GSS catalogs.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

binding: A request to include a particular column in a returned rowset. The binding specifies a property to be included in the search results.

bookmark: A marker that uniquely identifies a row within a set of rows.

catalog: The highest-level unit of organization in the Windows Search service. It represents a set of indexed documents against which queries can be executed by using the [MS-WSP].

category: A hierarchical grouping of rows. For example, a query result that contains author and title columns can be categorized based on author. Each group of rows containing the same value for author would constitute a category.

chapter: A range of rows within a set of rows.

column: The container for a single type of information in a row. Columns map to property names and specify what properties are used for the search query's command tree elements.

command tree: A combination of restrictions and sort orders that are specified for a search query.

cursor: An entity that is used as a mechanism to work with one row or a small block of rows (at one time) in a set of data returned in a result set. A cursor is positioned on a single row within the result set. After the cursor is positioned on a row, operations can be performed on that row or on a block of rows starting at that position.

Generic Search Service (GSS): A service that implements the back-end database functionality needed to provide results to search queries. The Windows Search Service is a Generic Search Service instance. The abstract interfaces that need to be implemented by a Generic Search Service are described in Local Events. These interfaces are called by the server in order to obtain appropriate responses to Windows Search Protocol messages.

Generic Security Services (GSS): An Internet standard, as described in [RFC2743], for providing security services to applications. It consists of an application programming interface (GSS-API) set, as well as standards that describe the structure of the security data.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the GUID. See also universally unique identifier (UUID).

handle: A token that can be used to identify and access cursors, chapters, and bookmarks.

hierarchical group coordinate: A coordinate that defines the position of a result in a hierarchical grouping result set. The coordinate is a list of non-negative integers, one per category as defined in the CPMCreateQueryIn message and CCategorizationSet structure, followed by another positive integer representing the position of the result in the last (deepest) category.

HRESULT: An integer value that indicates the result or status of an operation. A particular HRESULT can have different meanings depending on the protocol using it. See [MS-ERREF] section 2.1 and specific protocol documents for further details.

indexing: The process of extracting text or properties from files and storing the extracted values in an index or property cache.

inverted index: A persistent structure that contains the text content pulled out of files during indexing. The text in an inverted index maps from a word in a property to a list of the documents and locations within a document that contain that word.

locale: An identifier, as specified in [MS-LCID], that specifies preferences related to language. These preferences indicate how dates and times are to be formatted, how items are to be sorted alphabetically, how strings are to be compared, and so on.

named pipe: A named, one-way, or duplex pipe for communication between a pipe server and one or more pipe clients.

natural language query: A query constructed using human language instead of query syntax. The generic search service (GSS) is free to interpret the query in order to determine the best results. The interpretation is explicitly not specified in order to allow improvements over time.

noise word: A word that is ignored by the Windows Search service (WSS) when present in the restrictions specified for the search query, because it has little discriminatory value. English examples include "a," "and," and "the." Implementers of a generic search service (GSS) MAY choose to follow this guideline.

object: A file, email, email attachment, contact, calendar appointment or any other self-contained item that can be indexed and searched for by the GSS.

path: When referring to a file path on a file system, a hierarchical sequence of folders. When referring to a connection to a storage device, a connection through which a machine can communicate with the storage device.

property cache: A cache of file or object properties extracted during indexing.

restriction: A set of conditions that a file must meet to be included in the search results returned by the Generic Search Service (GSS) in response to a search query. A restriction narrows the focus of a search query, limiting the files that the Generic Search Service (GSS) will include in the search results only to those files matching the conditions.

row: The collection of columns containing the property values that describe a single result from the set of objects that matched the restrictions specified in the search query submitted to the Generic Search Service (GSS).

rowset: A set of rows returned in the search results.

sort order: A set of rules in a search query that defines the ordering of rows in the search result. Each rule consists of a managed property, such as modified date or size, and a direction for order, such as ascending or descending. Multiple rules are applied sequentially.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The Unicode standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

WHEREID: A 32-bit integer that uniquely identifies the query restriction. Cannot be equal to 0xFFFFFFFF.

Windows Search service (WSS): A service that creates indexed catalogs for the contents and properties of file systems. Applications can search the catalogs for information from the files on the indexed file system.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[IEEE754] IEEE, "IEEE Standard for Binary Floating-Point Arithmetic", IEEE 754-1985, October 1985, http://ieeexplore.ieee.org/servlet/opac?punumber=2355

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-ERREF] Microsoft Corporation, "Windows Error Codes".

[MS-LCID] Microsoft Corporation, "Windows Language Code Identifier (LCID) Reference".

[MS-SMB2] Microsoft Corporation, "Server Message Block (SMB) Protocol Versions 2 and 3".

[MS-SMB] Microsoft Corporation, "Server Message Block (SMB) Protocol".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[SALTON] Salton, G., "Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer", 1988, ISBN: 0201122278.

[UNICODE] The Unicode Consortium, "The Unicode Consortium Home Page", http://www.unicode.org/

1.2.2 Informative References

[Jones] Sparck Jones, K., Walker, S., and Robertson, S.E., "A Probabilistic Model of Information and Retrieval: Development and Status", September 1998, University of Cambridge Technical Report UCAM-CL-TR-446.

[MSDN-FULLPROPSPEC] Microsoft Corporation, "FULLPROPSPEC structure", http://msdn.microsoft.com/en-us/library/ms690996.aspx

[MSDN-OLEDBP] Microsoft Corporation, "OLE DB Provider for Indexing Service", http://msdn.microsoft.com/en-us/library/ms690319.aspx

[MSDN-PROPLIST] Microsoft Corporation, "Windows Properties", http://msdn.microsoft.com/en-us/library/windows/desktop/dd561977(v=VS.85).aspx

1.3 Overview

The WSS helps to efficiently organize the extracted features of a collection of documents. The Windows Search Protocol allows a client to communicate with a server hosting a GSS, both to issue queries and to allow an administrator to manage the indexing server.

When processing files, the WSS analyzes a set of documents, extracts useful information, and then organizes the extracted information in such a way that properties of those documents can be efficiently returned in response to queries. A collection of documents that can be queried comprises a catalog. A catalog can contain an inverted index (for quick word matching) and a property cache (for quick retrieval of property values).

Conceptually, a catalog consists of a logical "table" of properties with the text or value and corresponding locale stored in columns of the table. Each row of the table corresponds to a separate document in the scope of the catalog, and each column of the table corresponds to a property.

The specific tasks performed by the Windows Search Protocol are grouped into two functional areas:

- Remote administration of GSS catalogs
- Remote querying of GSS catalogs

1.3.1 Remote Administration Tasks

The Windows Search Protocol enables one GSS catalog management task from a client: querying the current state of a GSS catalog on the server (see CPMCiStateInOut).

All remote administration tasks follow a simple request/response model. No state is maintained on the client for any administration call, and administrative calls can be made in any order.

1.3.2 Remote Querying

The Windows Search Protocol enables clients to perform search queries against a remote server hosting a GSS.

Sending a search query is a multi-step process initiated by the client. The steps are as follows:

1. The client requests a connection to a server hosting a GSS.
2. The server verifies that the client is authorized and responds.
3. The client sends the parameters for the search query, which include:
    - Rowset properties like the catalog name and configuration information.
    - The restrictions to specify which documents are to be included and/or excluded from the search results.
    - The order in which the search results are to be returned.
    - The columns to be returned in the result set.
    - The maximum number of rows to be returned for the query.
    - The maximum time for query execution.
4. After the server has acknowledged the client's request to initiate the query, the client can request status information about the query, but this is not a required step.
5. The client then specifies which properties the server includes in the search results.

6. The client requests a result set from the server, and the server responds by sending the client the property values for files that were included in the results for the client's search query. If the value of a property is too large to fit in a single response buffer, the server will not send the property; instead, it will set the property status to deferred. The client then requests the property value separately using a series of requests for successive chunks of the value and then resumes requesting other values.

7. After the client is finished with the search query and no longer requires additional results, the client contacts the server to release the query.

8. After the server has released the query, the client may send a request to disconnect from the server. The connection is then closed. Alternatively, the client can issue another query and repeat the sequence from step 2.

1.4 Relationship to Other Protocols

The Windows Search Protocol relies on the SMB Protocol, as specified in [MS-SMB], for message transport. No other protocol depends directly on the Windows Search Protocol.

1.5 Prerequisites/Preconditions

The client is expected to have obtained the name of the server and a catalog name before this protocol is invoked. It is also assumed that the client and server have a security association usable with named pipes as specified in [MS-SMB].

1.6 Applicability Statement

The Windows Search Protocol is designed for querying and managing catalogs on a remote server from a client.

1.7 Versioning and Capability Negotiation

Windows Search Protocol has its own version ID that is used to communicate capabilities between the server and the client. There are two parts to this version ID: the version number, which is held in the least significant 2 bytes; and flags that are added to this number, comprising the remaining bytes. This version ID is added to the CPMConnectIn and CPMConnectOut messages, and can be used to check the versioning of the server or client throughout the transaction.

Starting with the release of Windows Search 4.0, messages include a checksum that is validated by the server. The version number of this is 0x109.

If the version number is greater than 0x109, this means that the clients include a checksum in all of their messages. Servers check all of the client's messages against the checksum if such a version is given.

0x10000 is added to the version ID and is used when identifying whether the operating system is 32-bit or 64-bit. Offsets are changed depending on this aspect of the architecture of the operating system.

Value	Meaning
0x00000102	32-bit Windows Server 2008 operating system, or 32-bit Windows Vista operating system.
0x00000109	32-bit Windows XP operating system, 32-bit Windows Server 2003 operating system, 32-bit Windows Home Server server software, 32-bit Windows Vista with Windows Search 4.0, 32-bit Windows Server 2003 with Windows Search 4.0. All of these versions of Windows are running Windows Search 4.0.

Value	Meaning
0x00000700	32-bit Windows 7 operating system.
0x00010102	64-bit version of Windows Vista or Windows Server 2008.
0x00010109	64-bit version of Windows Vista or Windows Server 2008 with Windows Search 4.0 installed.
0x00010700	64-bit Windows 7 or Windows Server 2008 R2 operating system.

1.8 Vendor-Extensible Fields

This protocol uses HRESULTs that are vendor-extensible. Vendors are free to choose their own values for this field, as long as the C bit (0x20000000) is set as specified in [MS-ERREF] section 2.1, indicating that the value is a customer code.

This protocol also uses NTSTATUS values taken from the NTSTATUS number space defined in [MS-ERREF]. Vendors SHOULD<1> reuse those values with their indicated meaning. Choosing any other value runs the risk of a collision in the future.

1.8.1 Property IDs

Properties are represented by IDs known as property IDs. Each property MUST have a GUID, as defined in [MS-DTYP] section 2.3.4.3. This identifier consists of a GUID representing a collection of properties called a property set plus either a string or a 32-bit integer to identify the property within the set. If the integer form of ID is used, then the values 0x00000000, 0xFFFFFFFF, and 0xFFFFFFFE are considered invalid.

Vendors can guarantee that their properties are uniquely defined by placing them in a property set defined by their own GUIDs.

1.9 Standards Assignments

This protocol has no standards assignments, only private assignments made by Microsoft using allocation procedures specified in other protocols.

Microsoft has allocated this protocol a named pipe as specified in [MS-SMB]. The pipe name is \pipe\MSFTEWDS.

2 Messages

The following sections specify how messages are transported and provide details of message syntax, including common structures and common error codes.

This protocol references commonly used data types as defined in [MS-DTYP].

2.1 Transport

All messages MUST be transported using a named pipe, as specified in [MS-SMB] or [MS-SMB2]. The following pipe name is used:

- \pipe\MSFTEWDS

This protocol uses the underlying server message block (SMB) named pipe protocol to retrieve the identity of the caller that made the connection as specified in [MS-SMB] section 2.2.4.9.1. The client MUST set SECURITY_IMPERSONATION as the **ImpersonationLevel** in the request to open the named pipe.<2>

2.2 Message Syntax

Several structures and messages in the following sections refer to chapter or bookmark handles. A handle is a 32-bit long opaque structure that uniquely identifies a chapter or bookmark. Typically, client applications receive handle values via method calls. However, there are several well-known values that need not be obtained from a server, the meaning of which is specified in the following table.

Value	Meaning
DB_NULL_HCHAPTER 0x00000000	A chapter handle to the unchaptered rowset that contains all query results.
DBBMK_FIRST 0xFFFFFFFC	A bookmark handle to a bookmark that identifies the first row in the rowset.
DBBMK_LAST 0xFFFFFFFD	A bookmark handle to a bookmark that identifies the last row in the rowset.

2.2.1 Structures

This section details data structures that are defined and used by the Windows Search Protocol.

All 2-byte, 4-byte, and 8-byte signed and unsigned integers in the following structures MUST be transferred in little-endian byte order.

The following table summarizes the data structures defined in this section.

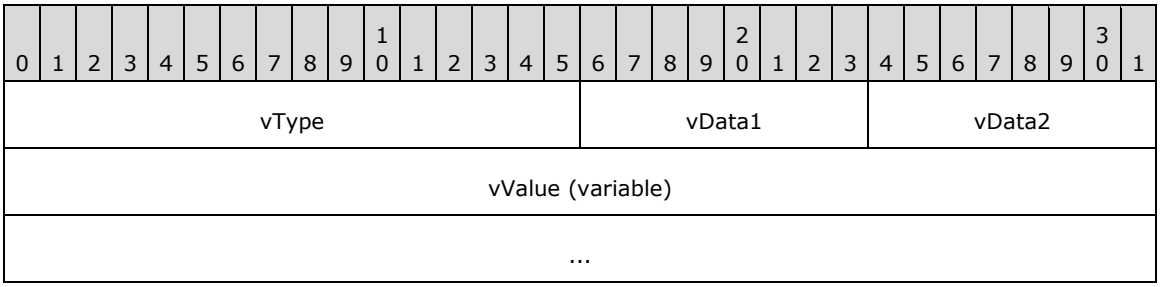
Structure	Description
CBaseStorageVariant	Contains the value on which to perform a match operation for a property that is specified in a CPropertyRestriction structure.
SAFEARRAY, SAFEARRAY2	Contains a multidimensional array.
SAFEARRAYBOUND	Represents the bounds for a dimension of an array specified in a SAFEARRAY

Structure	Description
	structure.
CFullPropSpec	Contains a property specification.
CContentRestriction	Contains a string to match for a property value in the property cache.
CInternalPropertyRestriction	Contains a property value to match with an operation.
CNatLanguageRestriction	Contains a natural language query match for a property.
CNodeRestriction	Contains an array of command tree nodes specifying the restrictions for a query.
CPropertyRestriction	Contains a property value to match with an operation.
CReuseWhere	Contains a WHEREID that refers to the restriction array.
CScopeRestriction	Contains a restriction on the files to be searched.
CSort	Identifies a column to sort.
CVectorRestriction	Contains an array of command tree nodes specifying the restrictions for a vector space array query (see [SALTON]).
CRestriction	A restriction node in a query command tree.
CRestrictionArray	Contains a counted array of restrictions.
CColumnSet	Describes the columns to return.
CCategorizationSet	A set of CCategorizationSpec structures where each describes the grouping property for one level in a hierarchical result set.
CCategorizationSpec	Specifies the categorization property used to categorize results at one level in a hierarchical result set.
CCategSpec	Contains a grouping specification.
CRangeCategSpec	Contains range information for grouping by ranges.
CDbColId	Contains a column identifier.
CDbProp	Contains a rowset property.
CDbPropSet	Contains a set of rowset properties.
CPidMapper	Maps from message internal property IDs (PIDs) to full property specifications.
CRowSeekAt	Contains the offset at which to retrieve rows in a CPMGetRowsIn message.
CRowSeekAtRatio	Identifies the approximate point expressed as a ratio at which to begin retrieval for a CPMGetRowsIn message.
CRowSeekByBookmark	Identifies the bookmarks from which to retrieve rows for a CPMGetRowsIn message.
CRowSeekNext	Contains the number of rows to skip in a CPMGetRowsIn message.
CRowsetProperties	Contains the configuration information for a query.
CSortSet	Contains the sort orders for a query.
CTableColumn	Contains a column for the CPMSetBindingsIn message.

Structure	Description
SERIALIZEDPROPERTYVALUE	Contains a serialized value.
CCoercionRestriction	Contains a coercion restriction that affects ranking of query results.
CFeedbackRestriction	Contains a set of feedback documents for relevance feedback queries (for more information, see [Jones]).
CProbRestriction	Contains probabilistic rank parameters (for more information, see [Jones]).
CRelDocRestriction	Contains a relevant document for relevance feedback queries (for more information, see [Jones]).
CAggregSet	Contains a set of aggregate specifications.
CAggregSpec	Contains a single aggregate or column specification.
CAggregSortKey	Contains a single grouping sort key.
CSortAggregSet	Contains a set of grouping sort keys.
CInGroupSortAggregSet	Contains sorting information for a group with regard to a parent group.
CInGroupSortAggregSets	Contains sorting information for a group with regard to one or more parent groups.
CCompletionCategSpec	Contains grouping information for building groups of completion suggestions.

2.2.1.1 CBaseStorageVariant

The CBaseStorageVariant structure contains the value on which to perform a match operation for a property specified in the CPropertyRestriction structure.



vType (2 bytes): A type indicator that indicates the type of **vValue**. It MUST be one of the values specified in the following table.

Value	Meaning
VT_EMPTY 0x0000	**vValue** is not present.
VT_NULL 0x0001	**vValue** is not present.
VT_I1 0x0010	A 1-byte signed integer.

Value	Meaning
VT_UI1 0x0011	A 1-byte unsigned integer.
VT_I2 0x0002	A 2-byte signed integer.
VT_UI2 0x0012	A 2-byte unsigned integer.
VT_BOOL 0x000B	A Boolean value; a 2-byte integer. **Note** Contains 0x0000 (FALSE) or 0xFFFF (TRUE).
VT_I4 0x0003	A 4-byte signed integer.
VT_UI4 0x0013	A 4-byte unsigned integer.
VT_R4 0x0004	An IEEE 32-bit floating point number, as defined in [IEEE754].
VT_INT 0x0016	A 4-byte signed integer.
VT_UINT 0x0017	A 4-byte unsigned integer. Note that this is identical to VT_UI4, except that VT_UINT cannot be used with VT_VECTOR (defined below); the value chosen is up to the higher layer that provides it to the Windows Search Protocol Specification, but the Windows Search Protocol Specification treats VT_UINT and VT_UI4 as identical with the exception noted above.
VT_ERROR 0x000A	A 4-byte unsigned integer containing an HRESULT, as specified in [MS-ERREF] section 2.1.
VT_I8 0x0014	An 8-byte signed integer.
VT_UI8 0x0015	An 8-byte unsigned integer.
VT_R8 0x0005	An IEEE 64-bit floating point number as defined in [IEEE754].
VT_CY 0x0006	An 8-byte two's complement integer (**vValue** divided by 10,000).
VT_DATE 0x0007	A 64-bit floating point number representing the number of days since 00:00:00 on December 31, 1899 (Coordinated Universal Time).
VT_FILETIME 0x0040	A 64-bit integer representing the number of 100-nanosecond intervals since 00:00:00 on January 1, 1601 (Coordinated Universal Time).
VT_DECIMAL 0x000E	A DECIMAL structure as specified in section 2.2.1.1.1.1.
VT_CLSID 0x0048	A 16-byte binary value containing a GUID.

Value	Meaning
VT_BLOB 0x0041	A 4-byte unsigned integer count of bytes in the blob, followed by that many bytes of data.
VT_BLOB_OBJECT 0x0046	A 4-byte unsigned integer count of bytes in the blob, followed by that many bytes of data.
VT_BSTR 0x0008	A 4-byte unsigned integer count of bytes in the string, followed by a string, as specified below under **vValue**.
VT_LPSTR 0x001E	A null-terminated string using the system code page.
VT_LPWSTR 0x001F	A null-terminated, 16-bit Unicode string. See [UNICODE]. **Note** The protocol uses UTF-16 LE encoding.
VT_COMPRESSED_LPWSTR 0x0023	A compressed version of a null-terminated, 16-bit Unicode string as specified in section 2.2.1.1.1.6. **Note** The protocol uses UTF-16 LE encoding.
VT_VARIANT 0x000C	CBaseStorageVariant.

The following table specifies the type modifiers for **vType**. Type modifiers can be binary OR'd with **vType** to change the meaning of **vValue** to indicate that it is one of two possible array types.

Value	Meaning
VT_VECTOR 0x1000	If the type indicator is combined with VT_VECTOR by using an OR operator, **vValue** is a counted array of values of the indicated type. See section 2.2.1.1.1.2. This type modifier MUST NOT be combined with the following types: VT_INT, VT_UINT, VT_DECIMAL, VT_BLOB, and VT_BLOB_OBJECT.
VT_ARRAY 0x2000	If the type indicator is combined with VT_ARRAY by an OR operator, the value is a SAFEARRAY containing values of the indicated type. This type modifier MUST NOT be combined with the following types: VT_I8, VT_UI8, VT_FILETIME, VT_CLSID, VT_BLOB, VT_BLOB_OBJECT, VT_LPSTR, and VT_LPWSTR.

vData1 (1 byte): When **vType** is VT_DECIMAL, the value of this field is specified as the **Scale** field in section 2.2.1.1.1.1. For all other **vTypes**, the value MUST be set to 0x00.

vData2 (1 byte): When **vType** is VT_DECIMAL, the value of this field is specified as the **Sign** field in section 2.2.1.1.1.1. For all other **vTypes**, the value MUST be set to 0x00.

vValue (variable): The value for the match operation. The syntax MUST be as indicated in the **vType** field.

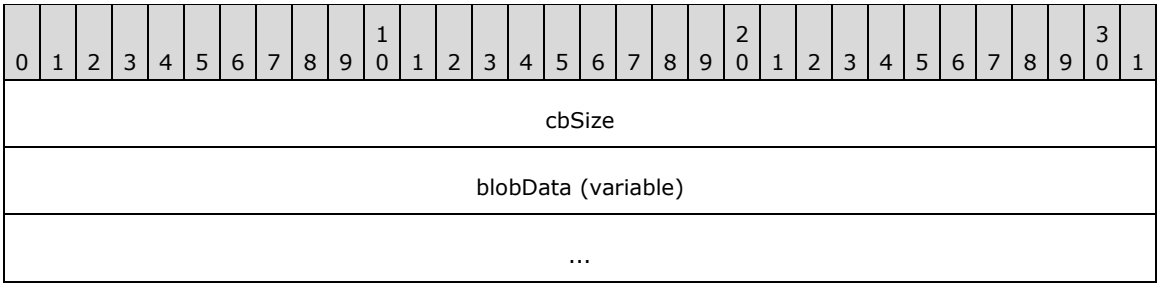
For fixed-length data types, the following table specifies the size of the **vValue** field in bytes.

vType	Size
VT_I1, VT,_UI1	1
VT_I2, VT_UI2, VT_BOOL	2
VT_I4, VT_UI4, VT_R4, VT_INT, VT_UINT, VT_ERROR	4
VT_I8, VT_UI8, VT_R8, VT_CY, VT_DATE, VT_FILETIME	8

vType	Size
VT_DECIMAL, VT_CLSID	16

For other types, the structure of **vValue** depends upon the value of **vType** as shown in the following table and diagrams.

Value of vType	Structure of vValue	Notes
VT_BLOB	First diagram.	
VT_BLOB_OBJECT	First diagram.	
VT_BSTR	First diagram.	
VT_LPSTR	Second diagram.	**string** is a null-terminated system code page string. **cLen** is the string's length in system code page characters.
VT_LPWSTR	Second diagram.	**String** is a null-terminated Unicode string. **cLen** is the string's length in Unicode characters.



cbSize (4 bytes): A 32-bit unsigned integer.

Note Indicates the size of the **blobData** field in bytes. If **vType** is set to VT_BSTR, cbSize MUST be set to 0x00000000 when the string represented is an empty string.

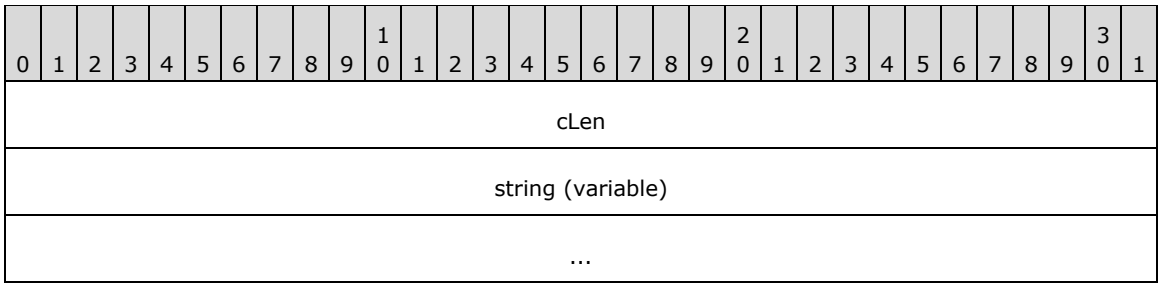
blobData (variable): MUST be of length **cbSize** in bytes.

For **vType** set to VT_BLOB or VT_BLOB_OBJECT, this field is opaque binary blob data.

For **vType** set to VT_BSTR, this field is a set of characters in an OEM–selected character set. The client and server MUST be configured to have interoperable character sets. There is no requirement that it be null-terminated.

For a **vType** set to either VT_LPSTR or VT_LPWSTR, the structure of **vValue** is shown in the diagram below, with the following caveats:

1. If **vType** is set to VT_LPSTR, then **cLen** indicates the size of the string in system code page characters and **string** is null-terminated.
2. If **vType** is set to VT_LPWSTR, then **cLen** indicates the size of the string in Unicode characters and **string** is a null-terminated Unicode string.



cLen (4 bytes): A 32-bit unsigned integer, indicating the size of the **string** field including the terminating null.

Note A value of 0x00000000 indicates that no such string is present.

string (variable): Null-terminated string.

Note This field MUST be absent if **cLen** equals 0x00000000.

2.2.1.1.1 CBaseStorageVariant Structures

The following structures are used in the CBaseStorageVariant structure.

2.2.1.1.1.1 DECIMAL

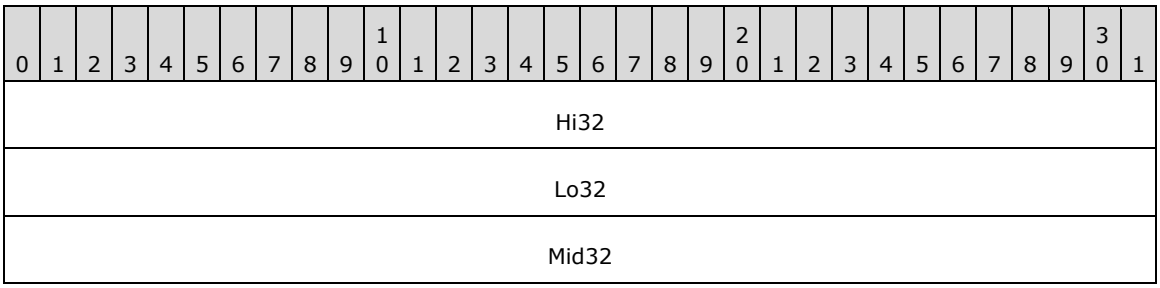
DECIMAL is used to represent an exact numeric value with a fixed precision and fixed scale.

When **vType** is set to VT_DECIMAL (0x0000E), the **vData1** and **vData2** fields of CBaseStorageVariant MUST be interpreted as follows:

vData1: The number of digits to the right of the decimal point. MUST be in the range 0 to 28.

vData2: The sign of the numeric value. Set to 0x00 if positive; set to 0x80 if negative.

When **vType** is set to VT_DECIMAL, the format of the **vValue** field is specified in the following diagram.



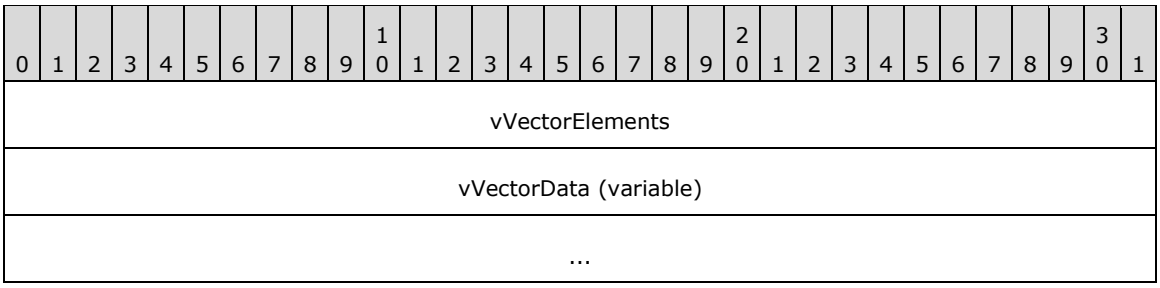
Hi32 (4 bytes): The highest 32 bits of the 96-bit integer.

Lo32 (4 bytes): The lowest 32 bits of the 96-bit integer.

Mid32 (4 bytes): The middle 32 bits of the 96-bit integer.

2.2.1.1.1.2 VT_VECTOR

VT_Vector is used to pass one-dimensional arrays.



vVectorElements (4 bytes): Unsigned 32-bit integer, indicating the number of elements in the **vVectorData** field.

vVectorData (variable): An array of items that have a type indicated by **vType** with the 0x1000 bit cleared. The size of an individual fixed-length item can be obtained from the fixed-length data type table, as specified in section 2.2.1.1. The length of this field, in bytes, can be calculated by multiplying **vVectorElements** by the size of an individual item.

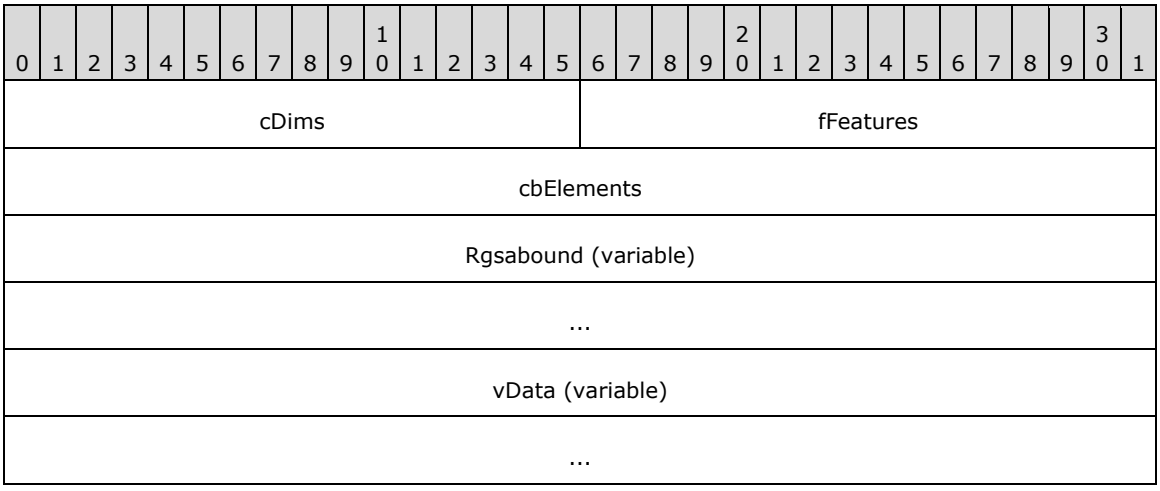
For variable-length data types, vVectorData contains a sequence of consecutively marshaled simple types where the type is indicated by **vType** with the 0x1000 bit cleared. This includes a special case indicated by **vType** VT_ARRAY | VT_VARIANT (that is, 0x100C).

The elements in the vVectorData field MUST be separated by 0 to 3 padding bytes such that each element begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this array. If padding bytes are present, the value they contain is arbitrary. The content of the padding bytes MUST be ignored by the receiver.

For a **vType** set to VT_ARRAY | VT_VARIANT, the type for items in this sequence is CBaseStorageVariant.

2.2.1.1.1.3 SAFEARRAY

SAFEARRAY is used to pass multidimensional arrays. The structure contains array size information as well as the data in the array.



cDims (2 bytes): Unsigned 16-bit integer, indicating the number of dimensions of the multidimensional array.

fFeatures (2 bytes): A 16-bit bitfield. The values represent features defined by upper-layer applications and MUST be ignored.

cbElements (4 bytes): A 32-bit unsigned integer specifying the size of each element of the array.

Rgsabound (variable): An array that contains one SAFEARRAYBOUND structure per dimension in the SAFEARRAY. This array has the left-most dimension first and the right-most dimension last.

vData (variable): A vector of marshaled items of a particular type, indicated by the **vType** of the containing CBaseStorageVariant, with the bit 0x2000 cleared.

vData is marshaled similarly to VT_Vector, as specified in section 2.2.1.1.1.2, but the number of items is not stored in front of the vector. Rather, the number of items is calculated by multiplying the **cElements** value with all safe array bounds given in the **Rgsabound** field. Elements are stored in a vector in order of dimensions, iterating beginning with the right-most dimension.

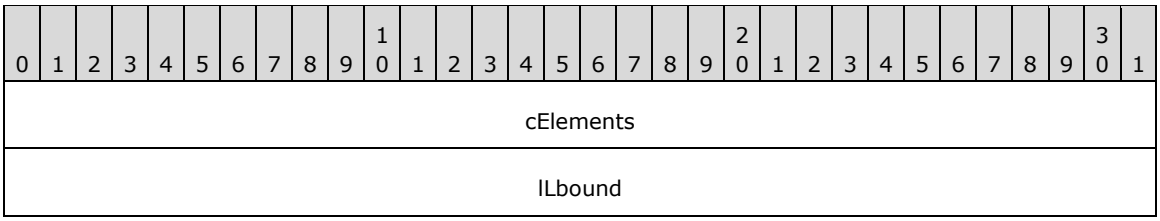
The following table visually represents a sample two-dimensional array. The first dimension has **cElements** equal to 4 (represented horizontally) and **lLbound** equal to 0. The second dimension has **cElements** equal to 2 (represented vertically) and **lLbound** equal to 0.

cElement 0	cElement 1	cElement 2	cElement 3
0x00000001	0x00000002	0x00000003	0x00000005
0x00000007	0x00000011	0x00000013	0x00000017

Using the previous diagram, vData will contain the following sequence: 0x00000001, 0x00000007, 0x00000002, 0x00000011, 0x00000003, 0x00000013, 0x00000005, 0x00000017 (iterating through the rightmost dimension first, and then incrementing the next dimension). The preceding **Rgsabound** (which records **cElements** and **lLbound**) would be: 0x00000004, 0x00000000, 0x00000002, and 0x00000000.

2.2.1.1.1.4 SAFEARRAYBOUND

The SAFEARRAYBOUND structure represents the bounds of one dimension of a SAFEARRAY or SAFEARRAY2. Its format is as follows.

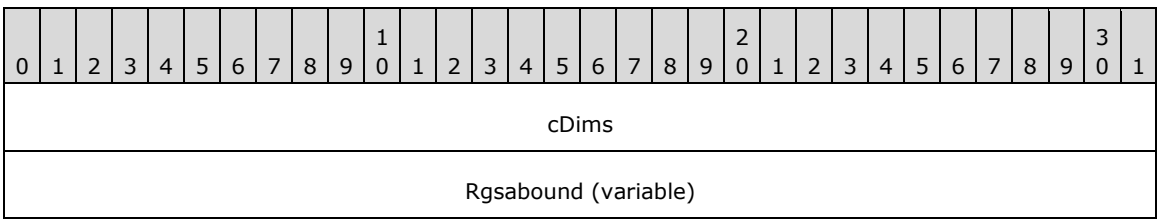


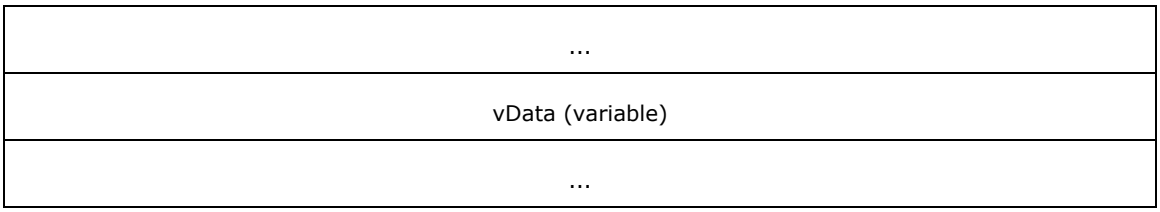
cElements (4 bytes): A 32-bit unsigned integer, specifying the number of elements in the dimension.

lLbound (4 bytes): A 32-bit unsigned integer, specifying the lower bound of the dimension.

2.2.1.1.1.5 SAFEARRAY2

SAFEARRAY2 is used to pass multidimensional arrays in SERIALIZEDPROPERTYVALUE. The structure contains boundary information as well as the data.





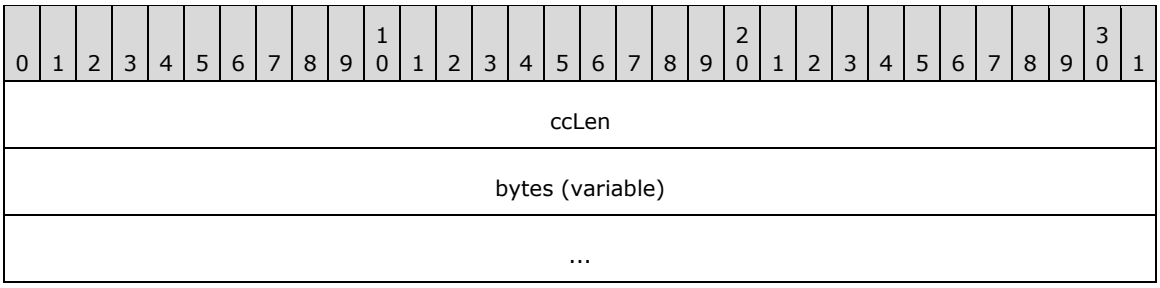
cDims (4 bytes): Unsigned 32-bit integer, indicating the number of dimensions of the SAFEARRAY2.

Rgsabound (variable): An array that contains one SAFEARRAYBOUND structure per dimension in the SAFEARRAY2. This array has the left-most dimension first and the right-most dimension last.

vData (variable): A vector of marshaled items of a particular type, indicated by the **dwType** of the containing SERIALIZEDPROPERTYVALUE, with bit 0x2000 cleared. The format of vData is the same as that specified for the vData field of SAFEARRAY.

2.2.1.1.1.6 VT_COMPRESSED_LPWSTR

The VT_COMPRESSED_LPWSTR structure contains a compressed version of a null-terminated, 16-bit Unicode string.

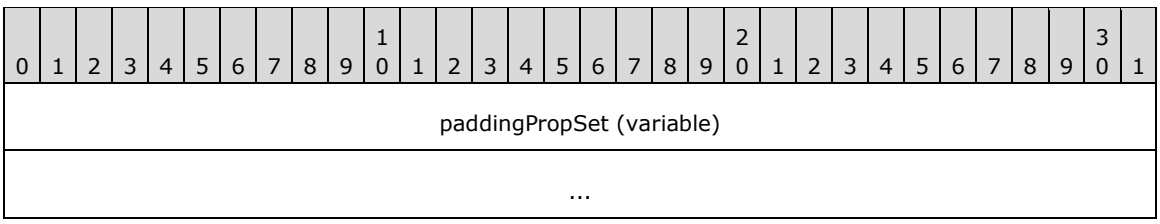


ccLen (4 bytes): A 32-bit unsigned integer, indicating the number of characters in the compressed Unicode string, excluding the terminating null character. A value of 0x00000000 indicates that no such string is present.

bytes (variable): A sequence of bytes, each representing the lower byte of a two-byte Unicode character, where the higher byte of the character is always set to zero. Note that only the first 255 Unicode characters can be represented with this encoding scheme. This field MUST be absent if **ccLen** is set to 0x00000000.

2.2.1.2 CFullPropSpec

The CFullPropSpec structure contains a property set GUID and a property identifier to uniquely identify a property. A CFullPropSpec instance has a property set GUID and either an integer property ID or a string property name. For properties to match, the CFullPropSpec structure MUST match the column identifier in the index. There is no conversion between property IDs and property names. Property names are case insensitive. (The **CFullPropSpec** structure corresponds to the **FULLPROPSPEC** structure described in [MSDN-FULLPROPSPEC].)



_guidPropSet (16 bytes)
...
...
ulKind
PrSpec
Property name (variable)
...

paddingPropSet (variable): This field MUST be 0 to 8 bytes in length. The length of this field MUST be such that the following field (_guidPropSet) begins at an offset that is a multiple of 8 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

_guidPropSet (16 bytes): The GUID of the property set to which the property belongs.

ulKind (4 bytes): A 32-bit unsigned integer. MUST be one of the following values that indicates the contents of **PrSpec**.

Value	Meaning
PRSPEC_LPWSTR 0x00000000	The **PrSpec** field specifies the number of non-null characters in the **Property name** field.
PRSPEC_PROPID 0x00000001	The **PrSpec** field specifies the property ID (PROPID).

PrSpec (4 bytes): A 32-bit unsigned integer with a meaning as indicated by the **ulKind** field.

Property name (variable): If **ulKind** is set to PRSPEC_PROPID, this field MUST NOT be present. If **ulKind** is set to PRSPEC_LPWSTR, this field MUST contain a case-insensitive array of **PrSpec** non-null Unicode characters that contains the name of the property.

2.2.1.3 CContentRestriction

The CContentRestriction structure contains a word or phrase to match in the inverted index for a specific property.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_Property (variable)																															
...																															
Padding1 (variable)																															

...
Cc
_pwcsPhrase (variable)
...
Padding2 (variable)
...
Lcid
_ulGenerateMethod

_Property (variable): A CFullPropSpec structure. This field indicates the property on which to perform a match operation.

Padding1 (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

Cc (4 bytes): A 32-bit unsigned integer, specifying the number of characters in the **_pwcsPhrase** field.

_pwcsPhrase (variable): A non-null-terminated Unicode string representing the word or phrase to match for the property. This field MUST NOT be empty. The **Cc** field contains the length of the string.

Padding2 (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

Lcid (4 bytes): A 32-bit unsigned integer, indicating the locale of **_pwcsPhrase**, as specified in [MS-LCID].

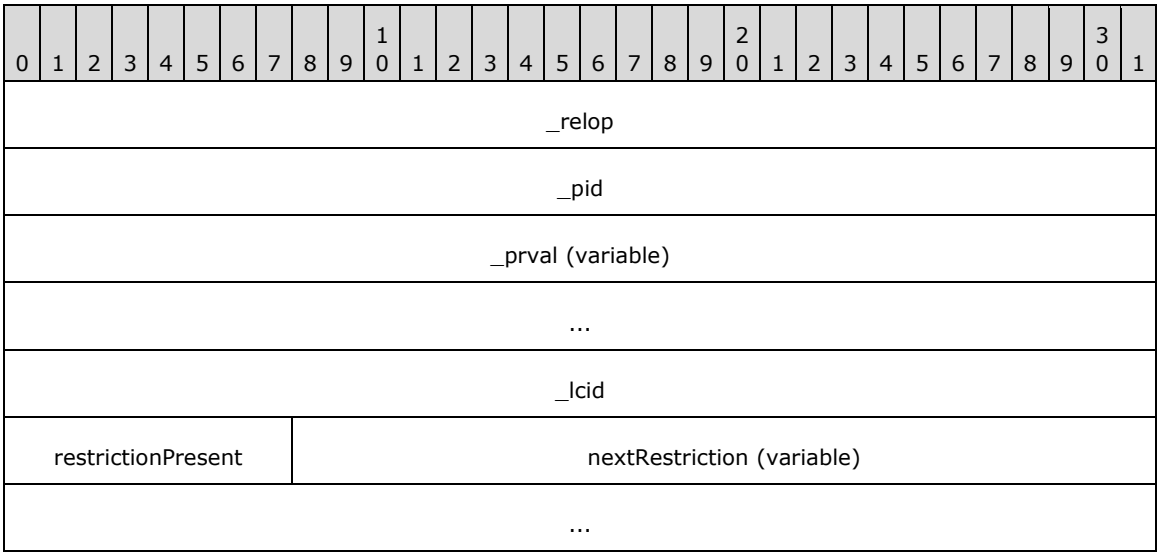
_ulGenerateMethod (4 bytes): A 32-bit unsigned integer, specifying the method to use when generating alternate word forms.

Value	Meaning
GENERATE_METHOD_EXACT 0x00000000	Exact match. Each word in the phrase matches exactly in the inverted index.
GENERATE_METHOD_PREFIX 0x00000001	Prefix match. Each word in the phrase is considered a match if the word is a prefix of an indexed string. For example, if the word "barking" is indexed, then "bar" would match when performing a prefix match.
GENERATE_METHOD_INFLECT 0x00000002	Matches inflections of a word. An inflection of a word is a variant of the root word in the same part of speech that has been modified, according to linguistic rules of a given language. For example, inflections of the verb swim in English include swim, swims, swimming, and swam. The inflection forms of a word can be determined by calling the **Inflect** abstract interface (section 3.1.7) to the GSS (Generic Search Service) with

Value	Meaning
	"*pwcsPhrase*" as an argument.

2.2.1.4 CInternalPropertyRestriction

The CInternalPropertyRestriction structure contains a property value to match with an operation.



_relop (4 bytes): A 32-bit integer specifying the relation to perform on the property. **_relop** MUST be one of the following values.

Value	Meaning
PRLT 0x00000000	A less-than comparison.
PRLE 0x00000001	A less-than or equal-to comparison.
PRGT 0x00000002	A greater-than comparison.
PRGE 0x00000003	A greater-than or equal-to comparison.
PREQ 0x00000004	An equality comparison.
PRNE 0x00000005	A not-equal comparison.
PRRE 0x00000006	A regular expression comparison.
PRAllBits 0x00000007	A bitwise AND that returns the right operand.

Value	Meaning
PRSomeBits 0x00000008	A bitwise AND that returns a nonzero value.
PRAll 0x00000100	The operation is to be performed on a column of a rowset and is only true if the operation is true for all rows.
PRAny 0x00000200	The operation is to be performed on a column of a rowset and is true if the operation is true for any row.

_pid (4 bytes): A 32-bit unsigned integer, representing the property ID.

_prval (variable): A CBaseStorageVariant that contains the value to relate to the property.

If the **vType** of _prval is VT_BLOB or VT_BLOB_OBJECT and _Property refers to a property of a string type (VT_LPSTR, VT_LPWSTR, VT_COMPRESSED_LPWSTR, VT_BSTR, or VT_VECTORs or VT_ARRAYs of those base types), then the first byte of the blob SHOULD be the low-order byte of a valid RANGEBOUNDARY ulType value (see section 2.2.1.23) and be followed by a null-terminated Unicode string. The value is interpreted in the same manner as RANGEBOUNDARY values (see section 2.2.1.23).

_lcid (4 bytes): A 32-bit unsigned integer, indicating the locale of a string, contained in **_prval** value, as specified in [MS-LCID].

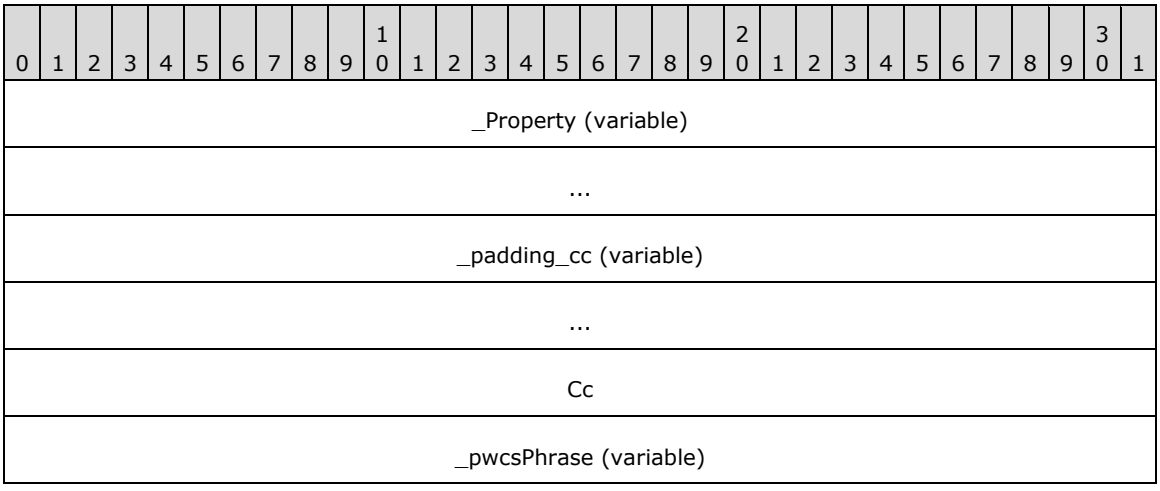
restrictionPresent (1 byte): A byte value. MUST be set to one of the following values.

Value	Meaning
0x00	**restrictionPresent** indicates that the **nextRestriction** field is not present.
0x01	**restrictionPresent** indicates that the **nextRestriction** field is present.

nextRestriction (variable): A CRestriction structure specifying a further restriction.

2.2.1.5 CNatLanguageRestriction

The CNatLanguageRestriction structure contains a natural language query match for a property. Natural language simply means that the string has no formal meaning. The GSS is free to match on the string in a variety of ways. It can drop words, add alternate forms, or make no changes.



...
_padding_lcid (variable)
...
Lcid

_Property (variable): A CFullPropSpec structure. This field indicates the property on which to perform the match operation.

_padding_cc (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

Cc (4 bytes): A 32-bit unsigned integer. The number of characters in the **_pwcsPhrase** field.

_pwcsPhrase (variable): A non-null-terminated Unicode string containing the text to search for within the specific property. MUST NOT be empty. The **Cc** field contains the length of the string.

_padding_lcid (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

Lcid (4 bytes): A 32-bit unsigned integer indicating the locale of **_pwcsPhrase**, as specified in [MS-LCID].

2.2.1.6 CNodeRestriction

The CNodeRestriction structure contains an array of command tree restriction nodes for constraining the results of a query.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_cNode																															
_paNode (variable)																															
...																															

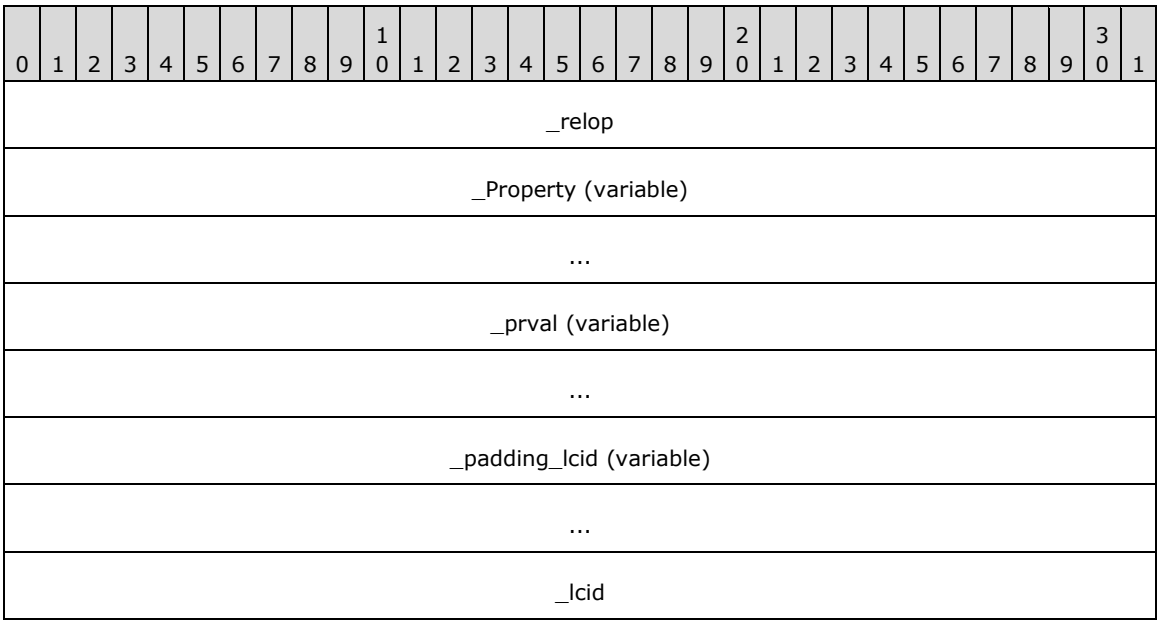
_cNode (4 bytes): A 32-bit unsigned integer specifying the number of CRestriction structures contained in the **_paNode** field.

_paNode (variable): An array of CRestriction structures. Structures in the array MUST be separated by 0 to 3 padding bytes such that each structure begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this array. If padding bytes are present, the value they contain is arbitrary. The content of the padding bytes MUST be ignored by the receiver.

2.2.1.7 CPropertyRestriction

The CPropertyRestriction structure contains a property to get from each row, a comparison operator and a constant. For each row, the value returned by the specific property in the row is compared

against the constant to determine if it has the relationship specified by the **_relop** field. For the comparison to be true, the datatypes of the values MUST match.



_relop (4 bytes): A 32-bit unsigned integer specifying the relation to perform on the property. **_relop** MUST be one of the following values.

Value	Meaning
PRLT 0x00000000	A less-than comparison.
PRLE 0x00000001	A less-than or equal-to comparison.
PRGT 0x00000002	A greater-than comparison.
PRGE 0x00000003	A greater-than or equal-to comparison.
PREQ 0x00000004	An equality comparison.
PRNE 0x00000005	A not-equal comparison.
PRRE 0x00000006	A regular expression comparison (see below).
PRAllBits 0x00000007	A bitwise AND that returns the value equal to **_prval**.
PRSomeBits 0x00000008	A bitwise AND that returns a nonzero value.

For vector properties, the behavior of the relational operators depends on the result of a logical **OR** using a mask and the relational operator.

If there is no mask, the restriction is true if the relational operator holds between each element of a property value and the corresponding element in the **_prval** field. If, in addition, the two vectors have different lengths, then the vector lengths are compared using the relational operator.

If there is a mask, its possible values are as follows.

Value	Meaning
PRAll 0x00000100	The restriction is true if every element in a property value has the relationship with some element in the **_prval** field.
PRAny 0x00000200	The restriction is true if any element in the property value has the relationship with some element in the **_prval** field.

For PRRE relations, regular expressions are expressed with a string that contains special symbols. Any character except an asterisk (*), period (.), question mark (?), or vertical bar (|) matches itself. A regular expression can be enclosed in matching quotes ("…"), and is enclosed in quotes if it contains a space or closing parenthesis (the ")" character).

The asterisk matches any number of characters. The period matches the end of the string. The question mark matches any one character. The vertical bar (|) is an escape character, which indicates special behavior for the characters in the table below the ([) character. The following table explains the meanings of special characters in regular expressions.

Character	Meaning
(	An opening parenthesis opens a group. It is followed by a matching closing parenthesis.
)	A closing parenthesis closes a group. It is preceded by a matching opening parenthesis.
[	An opening square bracket preceded (escaped) by a vertical pipe character opens a character class. It is followed by a matching (unescaped) closing square bracket.
{	An opening brace opens a counted match. It is followed by a matching closing brace.
}	A closing brace closes a counted match. It is preceded by a matching opening brace.
,	A comma separates OR clauses.
*	An asterisk matches zero or more occurrences of the preceding expression.
?	A question mark matches zero or one occurrence of the preceding expression.
+	A plus sign matches one or more occurrences of the preceding expression.
Other	All other characters match themselves.

The following table describes characters that, when located between square brackets ([ ]), have special meanings.

Character	Meaning
^	A caret matches everything but following classes. (It is the first character in the string.)
]	A closing square bracket matches another closing square bracket. It is preceded only by a caret (^); otherwise, it closes the class.

Character	Meaning
-	A hyphen is a range operator. It is preceded and followed by normal characters.
Other	All other characters match themselves (or begin or end a range).

The following table describes the syntax used between braces ({ }).

Character	Meaning
{m}	Matches exactly m occurrences of the preceding expression (0 < m < 256).
{m,}	Matches at least m occurrences of the preceding expression (1 < m < 256).
{m, n}	Matches between m and n occurrences of the preceding expression, inclusive (0 < m < 256, 0 < n < 256).

To match the asterisk and question mark, enclose them within brackets. For example, [*]sample matches "*sample".

_Property (variable): A CFullPropSpec structure indicating the property on which to perform a match operation.

_prval (variable): A CBaseStorageVariant structure containing the value to relate to the property.

If the **vType** of _prval is VT_BLOB or VT_BLOB_OBJECT, and _Property refers to a property of a string type (VT_LPSTR, VT_LPWSTR, VT_COMPRESSED_LPWSTR, VT_BSTR, or VT_VECTORs or VT_ARRAYs of those base types), then the first byte of the blob SHOULD be the low-order byte of a valid RANGEBOUNDARY ulType value (see section 2.2.1.23) and be followed by a null-terminated Unicode string. The value is interpreted in the same manner as RANGEBOUNDARY values (see section 2.2.1.23).

_padding_lcid (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following **_lcid** field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

_lcid (4 bytes): A 32-bit unsigned integer representing locale for the string contained in **_prval**, as specified in [MS-LCID].

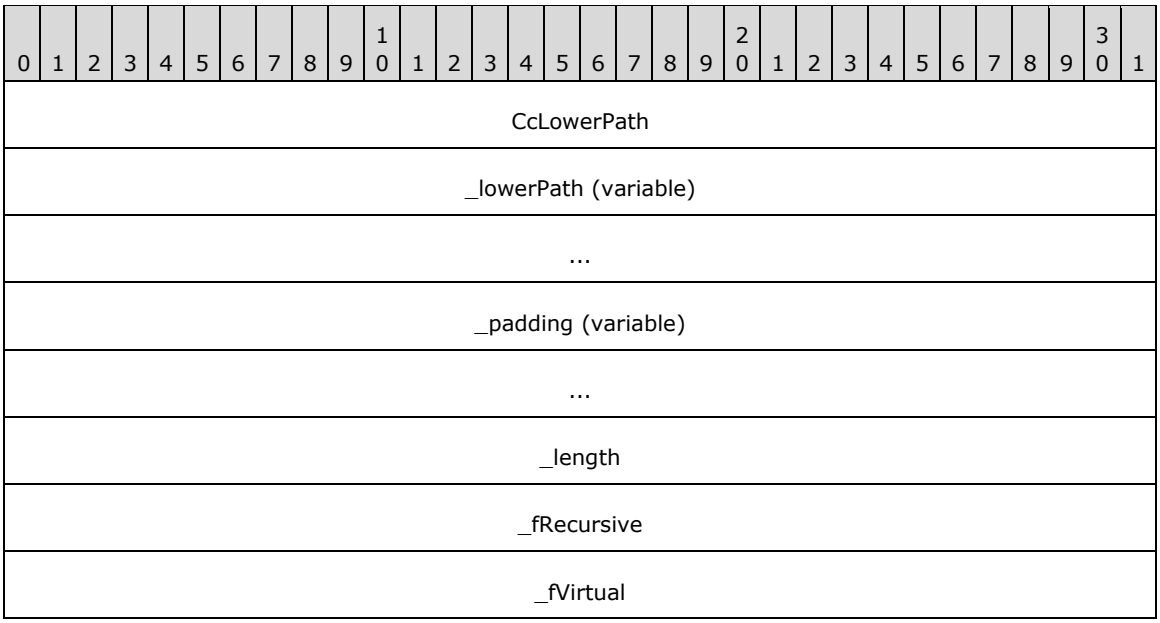
2.2.1.8 CReuseWhere

The CReuseWhere restriction packet contains a WHEREID that refers to the restriction array used to construct a currently open rowset. A rowset is open as long as there is still a cursor returned by CPMCreateQueryOut that has not been freed using CPMFreeCursorIn. A server can use this information to share evaluation of a restriction across multiple queries.

whereID (4 bytes): A 32-bit unsigned integer defining a unique WHEREID for referring to the CRestrictionArray.

2.2.1.9 CScopeRestriction

The CScopeRestriction structure restricts the files to be returned to those with a path that matches the restriction.



CcLowerPath (4 bytes): A 32-bit unsigned integer containing the number of Unicode characters in the **_lowerPath** field.

_lowerPath (variable): A non-null-terminated Unicode string representing the path to which the query is restricted. The **CcLowerPath** field contains the length of the string.

_padding (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

_length (4 bytes): A 32-bit unsigned integer containing the length of **_lowerPath** in Unicode characters. This MUST be the same value as **CcLowerPath**.

_fRecursive (4 bytes): A 32-bit unsigned integer. MUST be set to one of the following values:

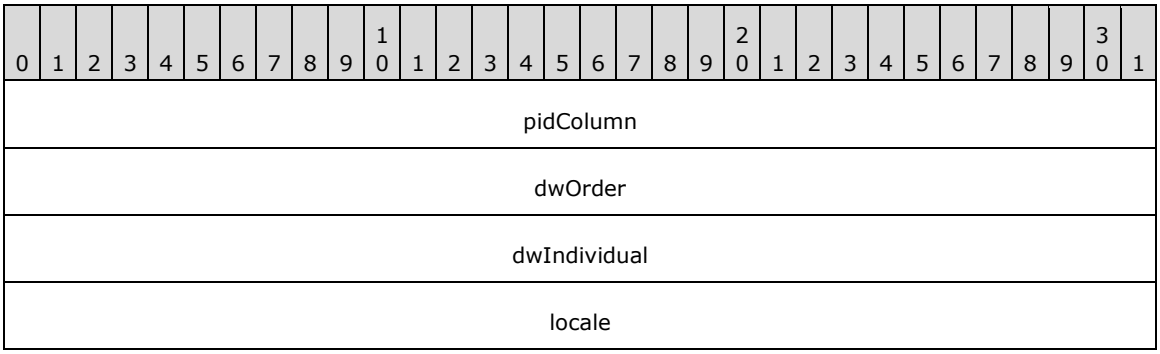
Value	Meaning
0x00000000	The server is not to examine any subdirectories.
0x00000001	The server is to recursively examine all subdirectories of the path.

_fVirtual (4 bytes): A 32-bit unsigned integer. MUST be set to one of the following values:

Value	Meaning
0x00000000	**_lowerPath** is a file system path.
0x00000001	**_lowerPath** is a virtual path (the URL associated with a physical directory on the file system) for a website.

2.2.1.10 CSort

The CSort structure identifies a column, direction, and locale to sort by.



pidColumn (4 bytes): A 32-bit unsigned integer. This is the index in CPidMapper for the property to sort by.

dwOrder (4 bytes): A 32-bit unsigned integer. MUST be one of the following values, specifying how to sort based on the column.

Value	Meaning
QUERY_SORTASCEND 0x00000000	The rows are to be sorted in ascending order based on the values in the column specified.
QUERY_DESCEND 0x00000001	The rows are to be sorted in descending order based on the values in the column specified.

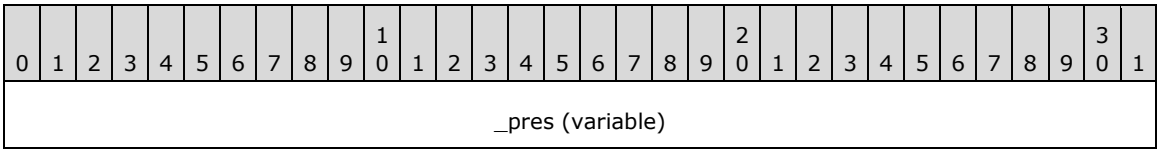
dwIndividual (4 bytes): A 32-bit unsigned integer. **dwIndividual** specifies how to treat properties of type VT_VECTOR with regard to sorting and MUST be one of the following values.

Value	Meaning
QUERY_SORTALL 0x00000000	The complete property is used for sorting, resulting in a single row for each result.
QUERY_SORTINDIVIDUAL 0x00000001	Each element of the VT_VECTOR is used for sorting independently, possibly resulting in multiple rows for a single result.

locale (4 bytes): A 32-bit unsigned integer indicating the locale (as specified in [MS-LCID]) of the column. The locale determines the sorting rules to use when sorting textual values. The GSS can use the appropriate operating system facilities to do this.

2.2.1.11 CVectorRestriction

The CVectorRestriction structure contains a weighted OR operation over restriction nodes. Vector restrictions represent queries using the full text vector space model of ranking (see [SALTON] for details). In addition to the OR operation, they also compute a rank based on the ranking algorithm.



...
_padding (variable)
...
_ulRankMethod

_pres (variable): A CNodeRestriction command tree upon which a ranked OR operation is to be performed.

_padding (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length is nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

_ulRankMethod (4 bytes): A 32-bit unsigned integer specifying a ranking algorithm. MUST be set to one of the following values.

Value	Meaning
VECTOR_RANK_MIN 0x00000000	Use the minimum algorithm, as specified in [SALTON].
VECTOR_RANK_MAX 0x00000001	Use the maximum algorithm, as specified in [SALTON].
VECTOR_RANK_INNER 0x00000002	Use the inner product algorithm, as specified in [SALTON].
VECTOR_RANK_DICE 0x00000003	Use the Dice coefficient algorithm, as specified in [SALTON].
VECTOR_RANK_JACCARD 0x00000004	Use the Jaccard coefficient algorithm, as specified in [SALTON].

2.2.1.12 CCoercionRestriction

The CCoercionRestriction structure contains the modifier and rank coercion operation.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_flValue																															
_childRes (variable)																															
...																															

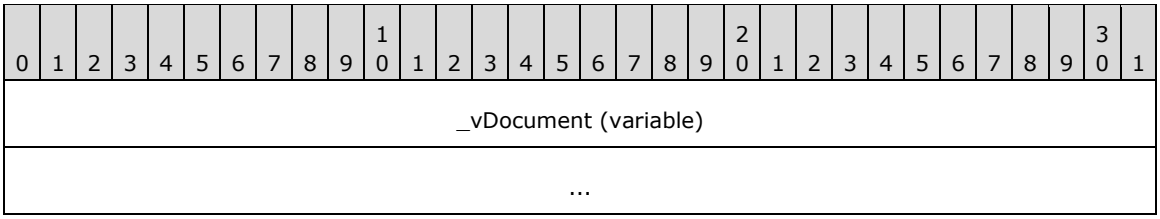
_flValue (4 bytes): An IEEE 32-bit floating point number [IEEE754] representing the coercion value upon which the rank coercion operation happens. Note that the coercion operation is determined by the containing CRestriction structure. The following coercion operations are supported.

Operation	Meaning
ADD	The rank value returned is the sum of the original rank value and the coercion value.
MULTIPLY	The rank value returned is the product of the original rank value and the coercion value and MUST be in the range 0.001 to 1.0.
ABSOLUTE	The rank value returned is the value specified in the coercion value and MUST be in the range 0 to 1000.

_childRes (variable): CRestriction structure that specifies a command tree. The returned rank value for results of a child restriction will be coerced as specified by the containing CRestriction structure.

2.2.1.13 CRelDocRestriction

A CRelDocRestriction structure contains a relevant document ID.



_vDocument (variable): A CBaseStorageVariant structure that specifies a relevant document for a relevance feedback query. The **vType** field of the **_vDocument** structure MUST be set to either VT_I4 or to VT_BSTR, specifying a URL string.

If **vType** is set to VT_I4, then **vValue** MUST be set to the document ID.

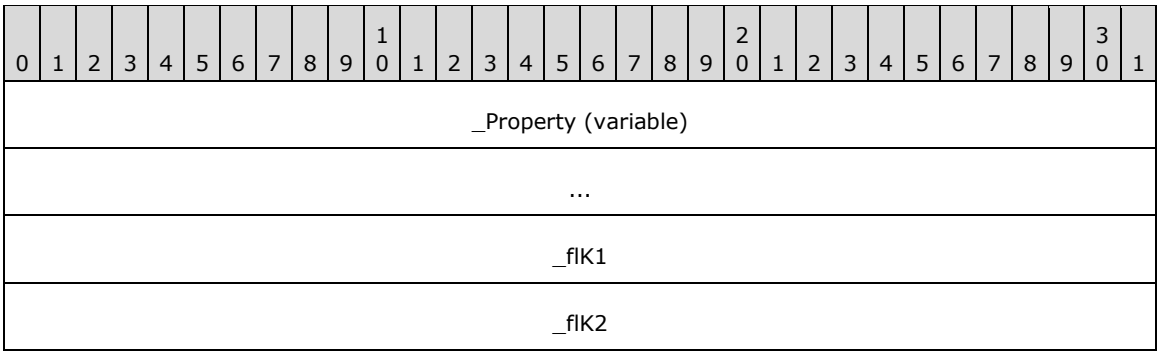
If **vType** is set to VT_BSTR, the URL string MUST be formatted by concatenating a constant string containing the docId protocol, the application, and the catalog name with the document ID in decimal representation, as in the following example.

```
docid:Windows.SystemIndex.153
```

In the example, "Windows" is the application name, "SystemIndex" is the catalog name, and "153" is the document ID for the document in decimal notation.<3>

2.2.1.14 CProbRestriction

A CProbRestriction structure contains parameters for probabilistic ranking.



_flK3
_flB
_cFeedbackDoc
_ProbQueryPid

_Property (variable): A CFullPropSpec structure, indicating which property to use for probabilistic ranking or the columns' group full property specification (which corresponds to **_groupPid** field in the CColumnGroup structure). In the latter case, CFullPropSpec MUST have the **_guidPropSet** field set to zero, the **ulKind** field set to PRSPEC_LPWSTR and the **Property Name** field set to the name of the referenced group property.

_flK1 (4 bytes): An IEEE 32-bit floating point number [IEEE754] that indicates parameter k1 in formula [1], specified below.

_flK2 (4 bytes): An IEEE 32-bit floating point number.

Note MUST be set to 0.0.

_flK3 (4 bytes): An IEEE 32-bit floating point number that indicates parameter k3 in formula [1].

_flB (4 bytes): An IEEE 32-bit floating point number that indicates parameter b in formula [1] below.

_cFeedbackDoc (4 bytes): A 32-bit unsigned integer specifying the count of relevant documents.

_ProbQueryPid (4 bytes): A 32-bit unsigned integer.

Note Reserved. MUST be set to 0x00000000.

Formula [1] for probabilistic ranking is the following sum for each query term:

$$\sum \frac{wtf(k_1+1)}{k_1\left((1-b)+b\frac{dl}{avdl}\right)+wtf} \times \log\left(\frac{N-n+0.5}{n+0.5}\right) \times \frac{qtf}{k_3+qtf}$$

Figure 1: Linear equation for probabilistic ranking restrictions in search request

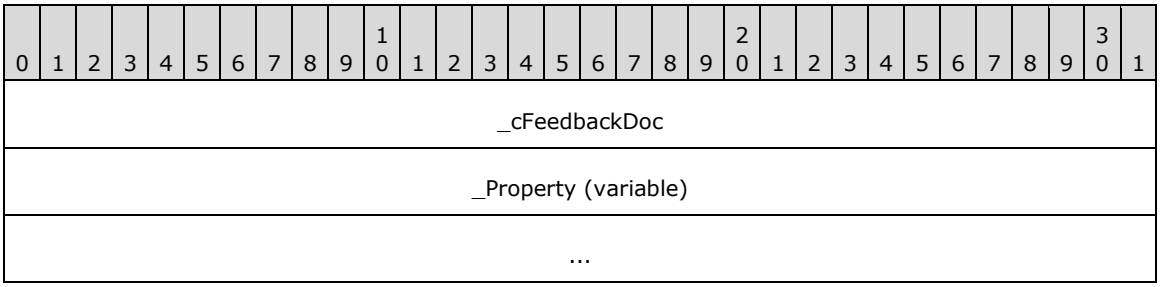
Where:

- wtf (weighted term frequency) is the sum of term frequencies (the number of times a term occurs in a document) of a given term multiplied by weights across all properties.
- dl is the document length, in terms of number of all words (including noise words).
- avdl is the average document length in the corpus, in terms of number of words (including noise words).
- N is the total number of documents in the corpus.
- n is the number of documents in the corpus that have the given query term.
- qtf is the number of documents containing the given query term; the sum is across all query terms.

- k1, k3, and b are parameters, specified in the CProbRestriction structure.

2.2.1.15 CFeedbackRestriction

The CFeedbackRestriction structure contains the number of relevant documents and a property specification for a relevance feedback query.

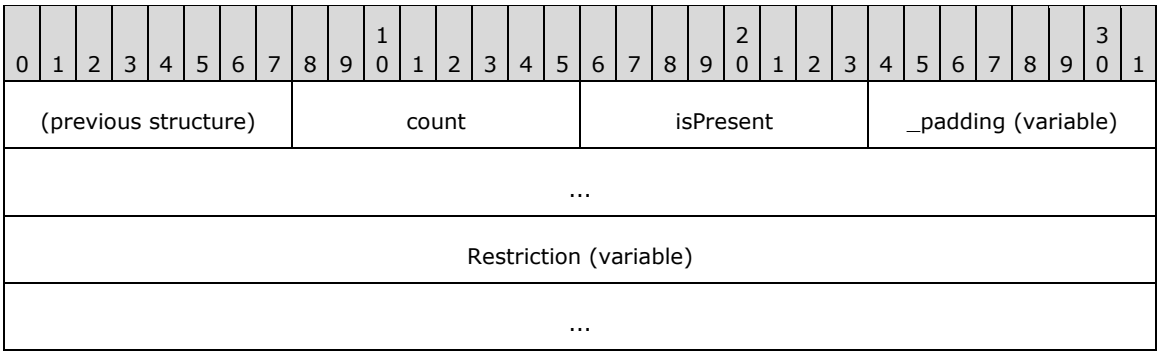


_cFeedbackDoc (4 bytes): A 32-bit unsigned integer specifying the count of relevant documents.

_Property (variable): A CFullPropSpec structure, specifying a property.

2.2.1.16 CRestrictionArray

The CRestrictionArray structure contains an array of restriction nodes. The first two fields (**count** and **isPresent**) are not padded and will start where the previous structure in the message ended (as indicated by the "previous structure" entry in the diagram below). The 1-byte length of "previous structure" is arbitrary and is not meant to suggest that **count** will begin on any particular boundary. However, the **Restriction** field MUST be aligned to begin at a multiple of 4 bytes from the beginning of the message, and hence the format is depicted as follows.



count (1 byte): An 8-bit unsigned integer specifying the number of **CRestriction** records contained in the **Restriction** field.

Note This field MUST be set to 0x01.

isPresent (1 byte): An 8-bit unsigned integer. MUST be set to one of the following values.

Value	Meaning
0x00	The **_padding** and **Restriction** fields are omitted.
0x01	The **_padding** and **Restriction** fields are present.

_padding (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the

message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

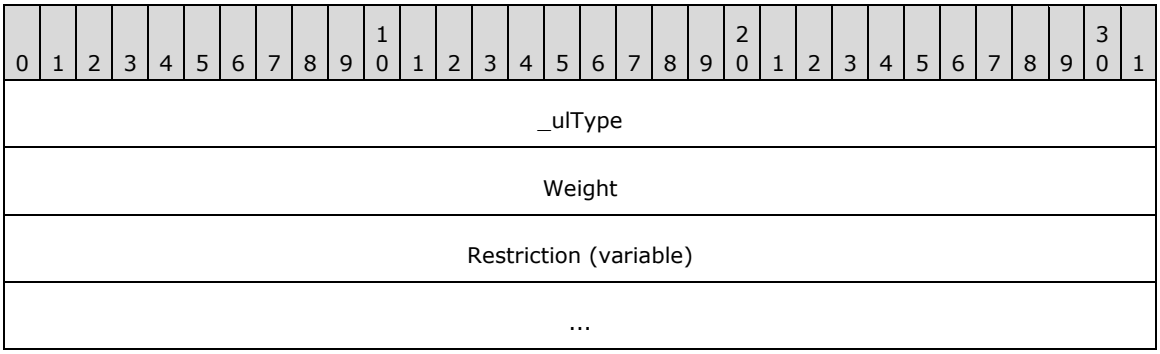
Note This field MUST be omitted if the value of **isPresent** is set to 0x00.

Restriction (variable): A CRestriction structure, specifying a node of a query command tree.

Note Restriction MUST be omitted if the value of **isPresent** is set to 0x00.

2.2.1.17 CRestriction

The CRestriction structure contains a restriction node in a query command tree.



_ulType (4 bytes): A 32-bit unsigned integer indicating the restriction type used for the command tree node. The type determines what is found in the **Restriction** field of the structure as described below. MUST be set to one of the following values.

Value	Meaning
RTNone 0x00000000	The node contains an empty **Restriction** (no value). It represents a node that would evaluate to no results. If under an RTNot node, the RTNot node would evaluate to the entire set of documents. If under an RTAnd node, the RTAnd node would also evaluate to no results. If under an RTOr node, the RTOr node would evaluate to whatever it would have evaluated to if the RTNone node was not there.
RTAnd 0x00000001	The node contains a CNodeRestriction on which a logical AND operation is to be performed. The CNodeRestriction contains the other restrictions that this operation is performed on.
RTOr 0x00000002	The node contains a CNodeRestriction on which a logical OR (disjunction) operation is to be performed. The CNodeRestriction contains the other restrictions that this operation is performed on.
RTNot 0x00000003	The node contains a CRestriction on which a NOT operation is to be performed.
RTContent 0x00000004	The node contains a CContentRestriction.
RTProperty 0x00000005	The node contains a CPropertyRestriction.
RTProximity 0x00000006	The node contains a CNodeRestriction with an array of CContentRestriction structures. Any other kind of restriction is undefined. The restriction requires the words or phrases found in the CContentRestriction structures to be within the GSS defined range in order to be a match. The WSS implementation computes a rank based on how far apart the words or phrases are. Implementations of the GSS can choose to do the same.

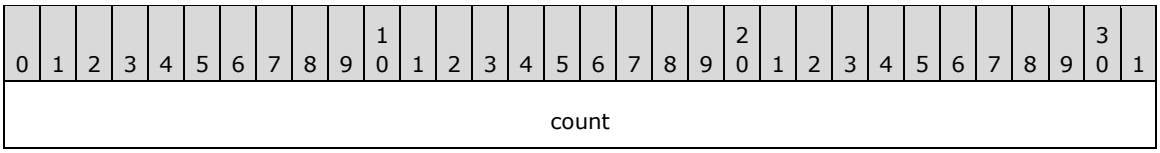
Value	Meaning
RTVector 0x00000007	The node contains a CVectorRestriction.
RTNatLanguage 0x00000008	The node contains a CNatLanguageRestriction.
RTScope 0x00000009	The node contains a CScopeRestriction.
RTReuseWhere 0x00000011	The node contains a CReuseWhere restriction.
RTInternalProp 0x00FFFFFA	The node contains a CInternalPropertyRestriction.
RTPhrase 0x00FFFFFD	The node contains a CNodeRestriction on which a phrase match is to be performed. The restrictions in the CNodeRestriction can only be a RTContent node. Otherwise, an error MUST be returned.
RTCoerce_Add 0x0000000A	The node contains a CCoercionRestriction structure with operation ADD, as specified in section 2.2.1.12.
RTCoerce_Multiply 0x0000000B	The node contains a CCoercionRestriction with structure operation MULTIPLY, as specified in section 2.2.1.12.
RTCoerce_Absolute 0x0000000C	The node contains a CCoercionRestriction structure with operation ABSOLUTE, as specified in section 2.2.1.12.
RTProb 0x0000000D	The node contains a CProbRestriction structure.
RTFeedback 0x0000000E	The node contains a CFeedbackRestriction structure.
RTReldoc 0x0000000F	The node contains a CRelDocRestriction structure.

Weight (4 bytes): A 32-bit unsigned integer representing the weight of the node. Weight indicates the node's importance relative to other nodes in the query command tree. This weight is used to calculate the rank of each node, although the exact effect of the weight is undefined. The guidance is that results from higher-weighted nodes usually return a higher rank than results from nodes that are the same but weighted lower. Implementers of the GSS can choose the exact algorithm.

Restriction (variable): The restriction type for the command tree node. The syntax MUST be as indicated by the **_ulType** field.

2.2.1.18 CColumnSet

The CColumnSet structure specifies the column numbers to be returned. This structure is always used in reference to a specific CPidMapper structure.



indexes (variable)
...

count (4 bytes): A 32-bit unsigned integer specifying the number of elements in the **indexes** array.

indexes (variable): An array of 4-byte unsigned integers each representing a zero-based index into the **aPropSpec** array in the corresponding CPidMapper structure. The corresponding property values are returned as columns in the result set.

2.2.1.19 CCategorizationSet

The CCategorizationSet structure contains information on how the grouping is done at each level in a hierarchical result set.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
count																															
categories (variable)																															
...																															

count (4 bytes): A 32-bit unsigned integer containing the number of elements in the **categories** array.

categories (variable): Array of CCategorizationSpec structures specifying the grouping for each level in a hierarchical query. The first structure specifies the top level.

2.2.1.20 CCategorizationSpec

The CCategorizationSpec structure specifies how grouping is done at one level in a hierarchical query.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_csColumns (variable)																															
...																															
_Spec (variable)																															
...																															
_AggregSet (variable)																															
...																															
_SortAggregSet (variable)																															
...																															

_InGroupSortAggregSets (variable)
...
_cMaxResults

_csColumns (variable): A CColumnSet structure indicating the columns to return at that level in a hierarchical result set.

_Spec (variable): A CCategSpec structure specifying the type of categorization and the column for the group.

_AggregSet (variable): A CAggregSet structure specifying aggregate information for the group.

_SortAggregSet (variable): A CSortAggregSet structure specifying default sorting for the group.

_InGroupSortAggregSets (variable): A CInGroupSortAggregSets structure specifying sorting for the group with regard to the parent group's range boundaries.

_cMaxResults (4 bytes): A 32-bit unsigned integer. Reserved.

Note MUST be set to 0x00000000.

2.2.1.21 CCategSpec

The CCategSpec structure contains information about which grouping to perform over query results.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_ulCategType																															
_sortKey (16 bytes)																															
...																															
...																															
CRangeCategSpec (variable)																															
...																															

_ulCategType (4 bytes): MUST be set to one of the following values that indicates the type of grouping to perform.

Value	Meaning
CATEGORIZE_UNIQUE 0x00000000	Unique categorization. Each unique value forms a category.
CATEGORIZE_RANGE 0x00000003	Range categorization. Ranges are explicitly specified in CRangeCategSpec.
CATEGORIZE_COMPLETION	Categorization by completion suggestions. All the parameters for specifying

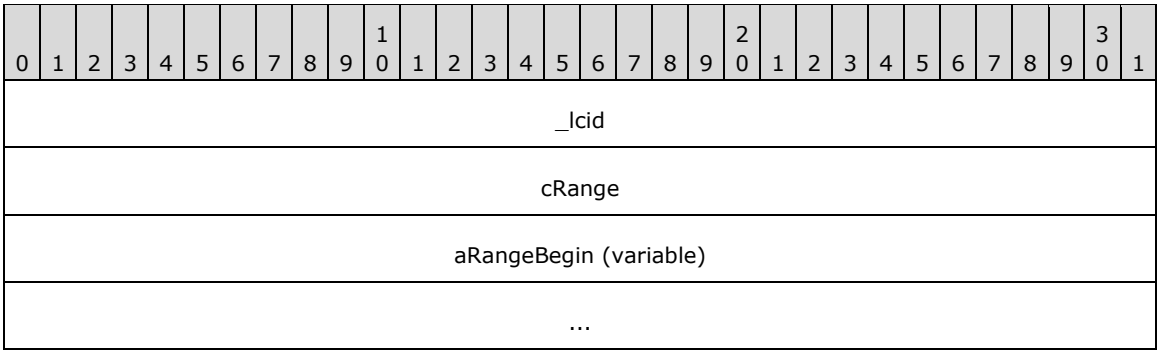
Value	Meaning
0x00000004	how these groups are built are in CCompletionCategSpec.

_sortKey (16 bytes): A CSort structure, specifying the sort order for the group.

CRangeCategSpec (variable): A CRangeCategSpec structure specifying the range values. This field MUST be omitted if **_ulCategType** is set to CATEGORIZE_UNIQUE; otherwise it MUST be present.

2.2.1.22 CRangeCategSpec

The CRangeCategSpec structure contains information about ranges for grouping into range-specified buckets.



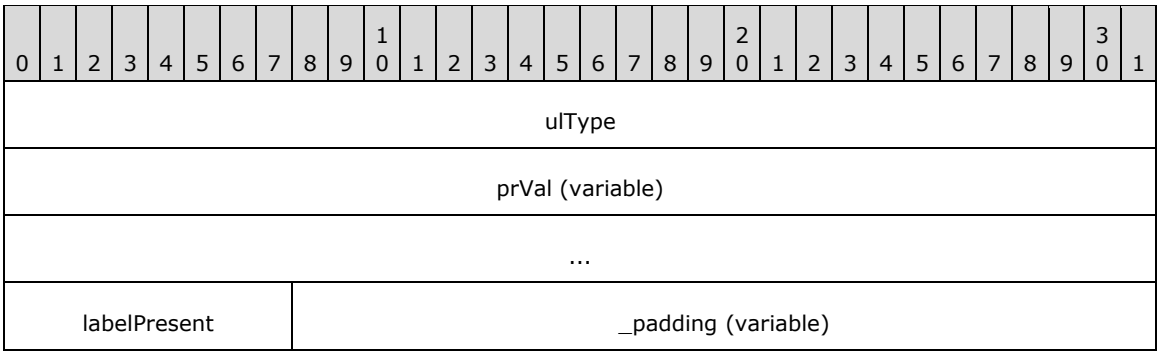
_lcid (4 bytes): A 32-bit unsigned integer. Reserved. This field can be set to any arbitrary value when sent.

cRange (4 bytes): A 32-bit unsigned integer, indicating the number of RANGEBOUNDARY structures in **aRangeBegin**.

aRangeBegin (variable): An array of RANGEBOUNDARY structures, specifying a set of ranges for which grouping is performed. Note that the first range is from minimum value to the boundary, represented by the first RANGEBOUNDARY structure. The next range is from where the first boundary cut off to the boundary represented by the second RANGEBOUNDARY structure, and so on. The last range includes all the items greater than the last RANGEBOUNDARY structure to the maximum value. There will be a total of **cRange** + 1 ranges. Values with **vType** set to VT_NULL and VT_EMPTY are always in the last group, regardless of sort order.

2.2.1.23 RANGEBOUNDARY

The RANGEBOUNDARY structure contains a single range.



...
ccLabel
Label (variable)
...

ulType (4 bytes): A 32-bit unsigned integer that indicates which type of boundary is represented by this structure.

Note MUST be set to one of the following values.

Value	Meaning
DBRANGEBOUNDTTYPE_BEFORE 0x00000000	MUST only be used for Unicode string values in **prVal**. Items with a value less than the Unicode string immediately preceding **prVal** lexicographically are included in the range.
DBRANGEBOUNDTTYPE_EXACT 0x00000001	Items with a value less than **prVal** are included in the range.
DBRANGEBOUNDTTYPE_AFTER 0x00000002	MUST only be used for Unicode string values in **prVal**. Items with a value less than the Unicode string immediately after **prVal** lexicographically are included in the range.

For example, two RANGEBOUNDARY structures of DBRANGEBOUND_EXACT with a **prVal** of "a" and DBRANGEBOUND_AFTER with a **prVal** of "z" could be used to partition the full Unicode range into three buckets:

```
1. x < "a"
2. "a" <= x <= "z"
3. x > "z"
```

prVal (variable): A CBaseStorageVariant structure.

Note Indicates the value for the range boundary. If **ulType** is set to DBRANGEBOUNDTTYPE_BEFORE or DBRANGEBOUNDTTYPE_AFTER, then the **vType** field of prVal MUST be set to a string type (VT_BSTR, VT_LPWSTR, or VT_COMPRESSED_LPWSTR).

labelPresent (1 byte): An 8-bit unsigned integer. MUST be set to one of the following values.

Value	Meaning
0x00	The **_padding**, **ccLabel**, and **Label** fields are omitted.
0x01	The **_padding**, **ccLabel**, and **Label** fields are present.

_padding (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length is nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

Note This field MUST be omitted if **labelPresent** is set to 0x00.

ccLabel (4 bytes): A 32-bit unsigned integer representing the number of characters in the **Label** field.

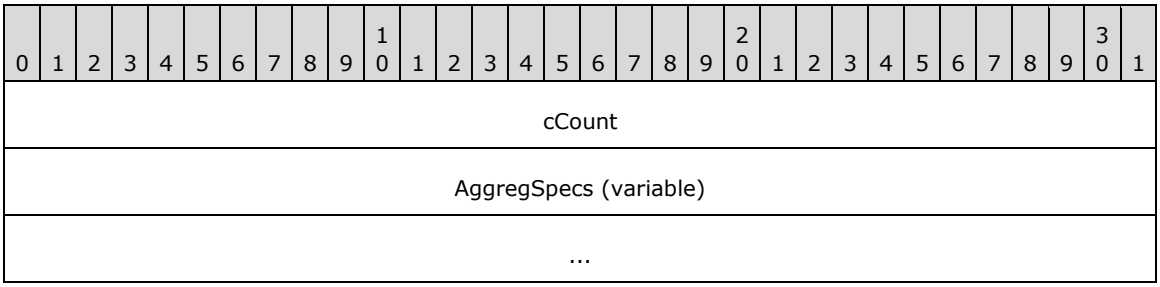
Note ccLabel MUST be omitted if **labelPresent** is set to 0x00; otherwise, it MUST be greater than zero.

Label (variable): A non-null-terminated Unicode string representing the label for this range. The **ccLabel** field contains the length of the string.

Note Label MUST be omitted if **labelPresent** is set to 0x00; otherwise, it MUST NOT be empty.

2.2.1.24 CAggregSet

The CAggregSet structure contains information about aggregates. Aggregate is a database concept for a field calculated using the information retrieved from the query. The different aggregates that are supported by GSS are defined in the CAggregSpec's type field (specified in section 2.2.1.25).

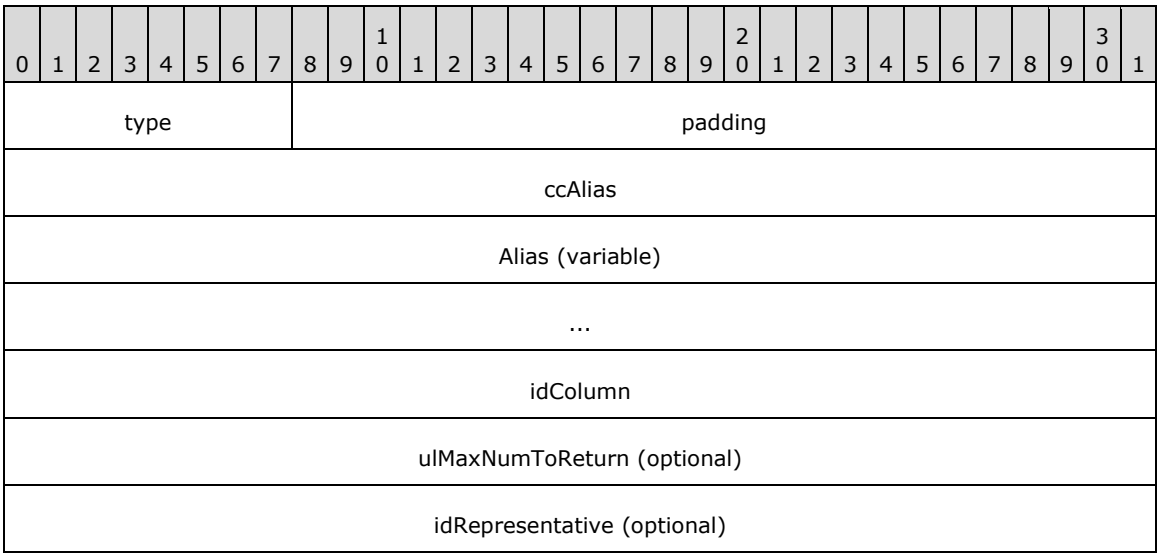


cCount (4 bytes): A 32-bit unsigned integer specifying the number of entries in **AggregSpecs**.

AggregSpecs (variable): An array of CAggregSpec structures, each describing individual aggregation.

2.2.1.25 CAggregSpec

The CAggregSpec structure contains information about an individual aggregate.



type (1 byte): An 8-bit unsigned integer specifying the type of aggregation used. The type MUST be one of the following values.

Value	Meaning
DBAGGTTYPE_BYNONE 0x00	No aggregation is used.
DBAGGTTYPE_SUM 0x01	Sum of the **idColumn** property value from each result in the group. Valid only for numeric properties.
DBAGGTTYPE_MAX 0x02	Maximum value of the **idColumn** property value from each result in the group. Valid only for numeric or filetime properties.
DBAGGTTYPE_MIN 0x03	Minimum value of the **idColumn** property value from each result in the group. Valid only for numeric or filetime properties.
DBAGGTTYPE_AVG 0x04	Average value of the **idColumn** property value from each result in the group. Valid only for numeric properties.
DBAGGTTYPE_COUNT 0x05	Count of the number of leaf results in the group.
DBAGGTTYPE_CHILDCOUNT 0x06	Count of immediate children of the group.
DBAGGTTYPE_BYFREQ 0x07	Most frequent N **idColumn** values from the results in the group. Additionally includes a count for how many times each value occurred and a document identifier for a result that has each returned value.
DBAGGTTYPE_FIRST 0x08	First N **idColumn** values from leaf results found in a group.
DBAGGTTYPE_DATERANGE 0x09	Lower and upper bounds of the **idColumn** values found in the group results group. Only valid for filetime properties.
DBAGGTTYPE_REPRESENTATIVEOF 0x0a	N **idRepresentative** values, each selected from one of the result subsets that have a unique **idColumn** value. Each value is also returned with a document identifier that has the idRepresentative value.
DBAGGTTYPE_EDITDISTANCE 0x0b	Edit distance between the results in a completion group and the primary query string in the Completions grouping clause, as specified in CCompletionCategSpec.

padding (3 bytes): This field MUST be 3 bytes in length, and the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

ccAlias (4 bytes): A 32-bit unsigned integer specifying the number of characters in the **Alias** field.

Alias (variable): A non-null-terminated Unicode string that represents the alias name for the aggregate. The **ccAlias** field contains the length of the string.

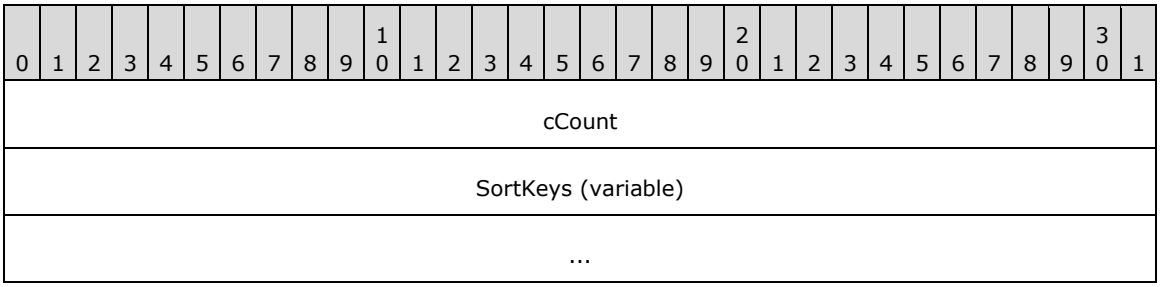
idColumn (4 bytes): Property ID for the column to be aggregated over.

ulMaxNumToReturn (4 bytes): An optional 32-bit unsigned integer that is the number of elements to return for **First**, **ByFreq**, and **RepresentativeOf** aggregates.

idRepresentative (4 bytes): An optional 32-bit unsigned integer that is the representative property ID requested for the **RepresentativeOf** aggregate.

2.2.1.26 CSortAggregSet

The CSortAggregSet structure contains information about group sorting.

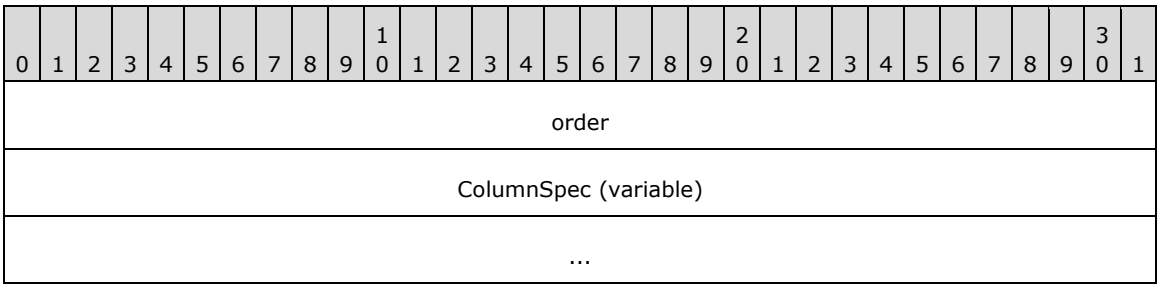


cCount (4 bytes): A 32-bit unsigned integer specifying the number of entries in **SortKeys**.

SortKeys (variable): An array of CAggregSortKey structures, each describing a sort order.

2.2.1.27 CAggregSortKey

The CAggregSortKey structure contains information about sort order over single column.



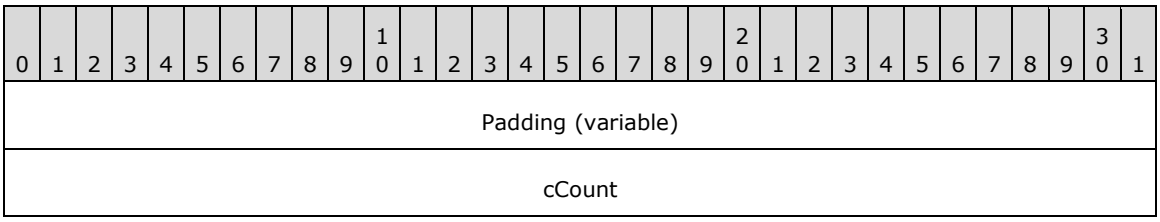
order (4 bytes): A 32-bit unsigned integer specifying sort order. MUST be set to one of the following values.

Value	Meaning
QUERY_SORTASCEND 0x00000000	The rows are to be sorted in ascending order based on the values in the column specified.
QUERY_DESCEND 0x00000001	The rows are to be sorted in descending order based on the values in the column specified.

ColumnSpec (variable): A CAggregSpec structure specifying which column to sort by.

2.2.1.28 CInGroupSortAggregSets

The CInGroupSortAggregSets structure contains information on how the group is sorted with regard to the parent's group ranges.



Reserved
SortSets (variable)
...

Padding (variable): Aligned to a 4-byte boundary.

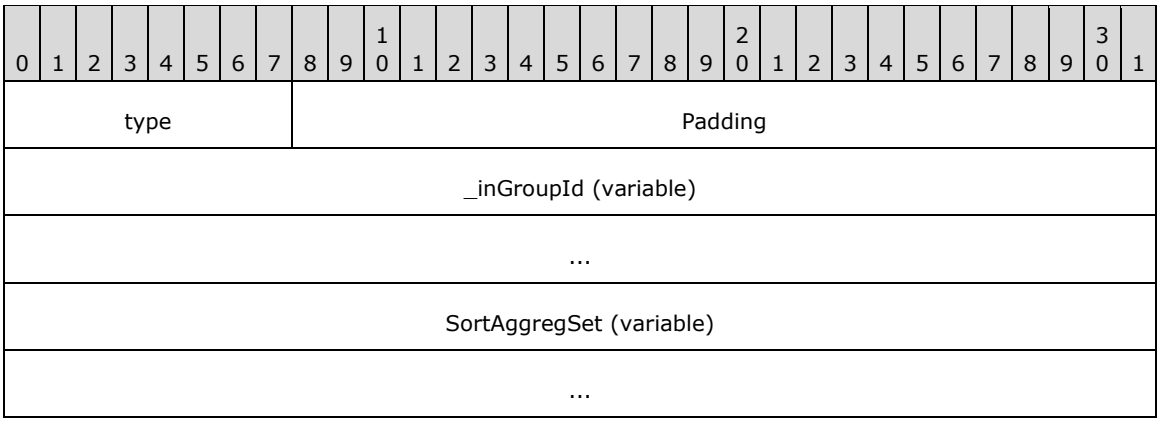
cCount (4 bytes): A 32-bit unsigned integer specifying the number of entries in **SortSets**.

Reserved (4 bytes): A 4-byte field that must be ignored.

SortSets (variable): An array of CSortSet structures.

2.2.1.29 CInGroupSortAggregSet

A CInGroupSortAggregSet structure specifies a sort order for a single range in the parent's group.



type (1 byte): An 8-bit unsigned integer specifying the type of range in the parent's group. **type** MUST be set to one of the following values.

Value	Meaning
GroupIdDefault 0x00	The default for all ranges.
GroupIdMinValue 0x01	The first range in the parent's group.
GroupIdNull 0x02	The last range in the parent's group.
GroupIdValue 0x03	The range with a particular range boundary value in the parent's group.

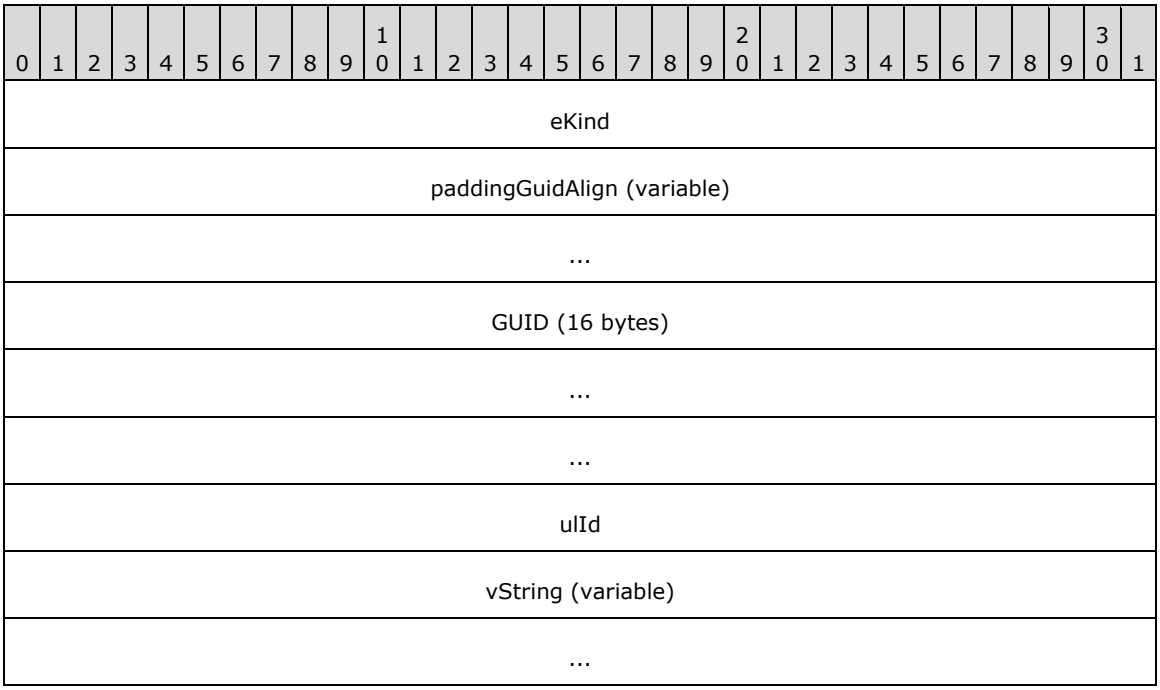
Padding (3 bytes): This field MUST be 3 bytes in length and the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

_inGroupId (variable): A CBaseStorageVariant structure that contains a value representing a range in the parent's group. This field MUST not be present if type is not equal to GroupIdValue.

SortAggregSet (variable): A CSortSet structure, specifying the sort order for the range in the parent's group.

2.2.1.30 CDbColId

The CDbColId structure contains a column identifier.



eKind (4 bytes): MUST be set to one of the following values that indicates the contents of GUID and **vValue**.

Value	Meaning
DBKIND_GUID_NAME 0x00000000	**vString** contains a property name.
DBKIND_GUID_PROPID 0x00000001	**ulId** contains a 4-byte integer indicating the property ID.

paddingGuidAlign (variable): This field MUST be 0 to 8 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 8 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

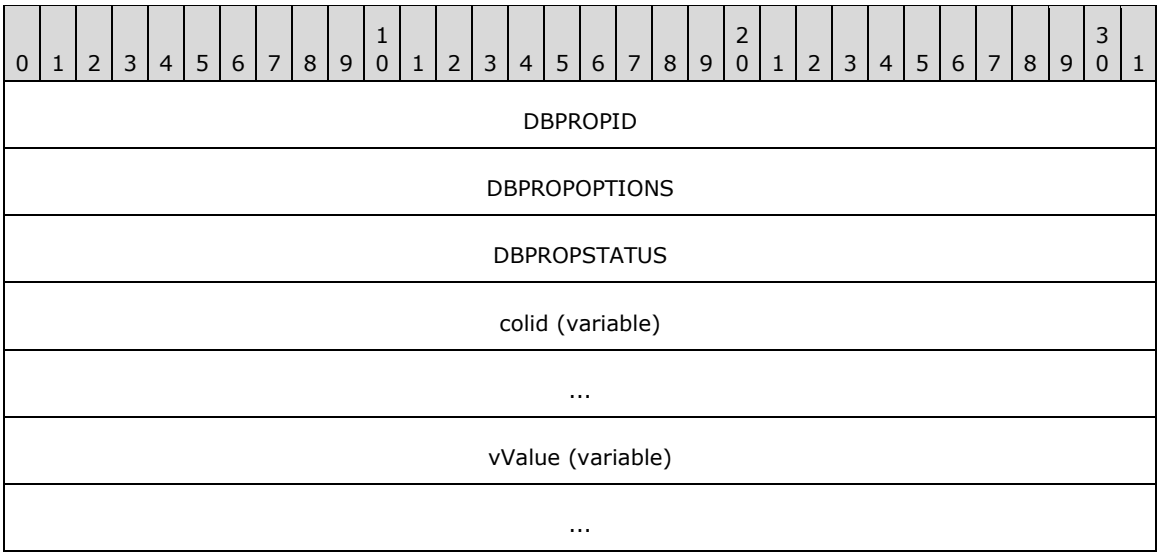
GUID (16 bytes): The property GUID.

ulId (4 bytes): If **eKind** is DBKIND_GUID_PROPID, this field contains an unsigned integer specifying the property ID. If **eKind** is DBKIND_GUID_NAME, this field contains an unsigned integer specifying the number of Unicode characters contained in the **vString** field.

vString (variable): A non-null-terminated Unicode string representing the property name. It MUST be omitted unless the **eKind** field is set to DBKIND_GUID_NAME.

2.2.1.31 CDbProp

The CDbProp structure contains a database property. These properties control how queries are interpreted by the GSS.



DBPROPID (4 bytes): A 32-bit unsigned integer indicating the property ID. This field uniquely identifies each property in a particular query, but has no other interpretation.

DBPROPOPTIONS (4 bytes): Property options. This field MUST be set to 0x00000001 if the property is optional and to 0x00000000 otherwise.

DBPROPSTATUS (4 bytes): Property status.

Note DBPROPSTATUS MUST be set to 0x00000000.

colid (variable): A CDbColId structure that defines the database property being passed.

vValue (variable): A CBaseStorageVariant containing the property value.

2.2.1.31.1 Database Properties

This section details the properties that are used by the Windows Search Protocol to control the behavior of the GSS. These properties are grouped into three property sets, identified in the **guidPropertySet** field of the CDbPropSet structure.

The following table lists the properties that are part of the DBPROPSET_FSCIFRMWRK_EXT property set.

Value	Meaning
DBPROP_CI_CATALOG_NAME 0x00000002	Specifies the name of the catalog or catalogs to query. The value MUST be a VT_LPWSTR or a VT_VECTOR | VT_LPWSTR.
DBPROP_CI_INCLUDE_SCOPES 0x00000003	Specifies one or more paths to be included in the query. The value MUST be a VT_LPWSTR or a VT_VECTOR | VT_LPWSTR.
DBPROP_CI_SCOPE_FLAGS 0x00000004	Specifies how the paths specified by the DBPROP_CI_INCLUDE_SCOPES property are to be treated. The value MUST be a VT_I4 or a VT_VECTOR | VT_I4.

Value	Meaning
DBPROP_CI_QUERY_TYPE 0x00000007	Specifies the type of query using a CDbColId structure. The structure MUST be set such that the **eKind** field contains 0x00000001 and the GUID and ulID fields are filled with zeros.

The following table lists the flags for the DBPROP_CI_SCOPE_FLAGS property.

Value	Meaning
QUERY_DEEP 0x01	If set, indicates that files in the scope directory and all subdirectories are included in the results. If clear, only files in the scope directory are included in the results.
QUERY_VIRTUAL_Path 0x02	If set, indicates that the scope is a virtual path. If clear, indicates that the scope is a physical directory.

The following table lists the query types for the DBPROP_CI_QUERY_TYPE property.

Value	Meaning
CiNormal 0x00000000	A regular query.

The following table lists the properties that are part of the DBPROPSET_QUERYEXT property set.

Value	Meaning
DBPROP_USECONTENTINDEX 0x00000002	Use the inverted index to optimize the speed of evaluating content restrictions at the cost of the index possibly being out of date. The value MUST be a VT_BOOL. If TRUE, the server is allowed to fail these queries.
DBPROP_DEFERNONINDEXEDTRIMMING 0x00000003	Some operations, such as filtering by scope or security, can be expensive. This flag indicates that it is acceptable to defer this filtering until the results are actually requested. The value MUST be a VT_BOOL.
DBPROP_USEEXTENDEDDBTYPES 0x00000004	Indicates if the client supports VT_VECTOR data types. If TRUE, the client supports VT_VECTOR; if FALSE, the server is to convert VT_VECTOR data types to VT_ARRAY data types. The value MUST be a VT_BOOL.
DBPROP_FIRSTROWS 0x00000007	If TRUE, the GSS returns the first rows that match. If FALSE, then rows by default are returned in order of descending rank. The value MUST be a VT_BOOL.
DBPROP_ENABLEROWSETEVENTS 0x00000010	If TRUE, this indicates that the server generates rowset events that are relevant to the associated query.This value MUST be a VT_BOOL.

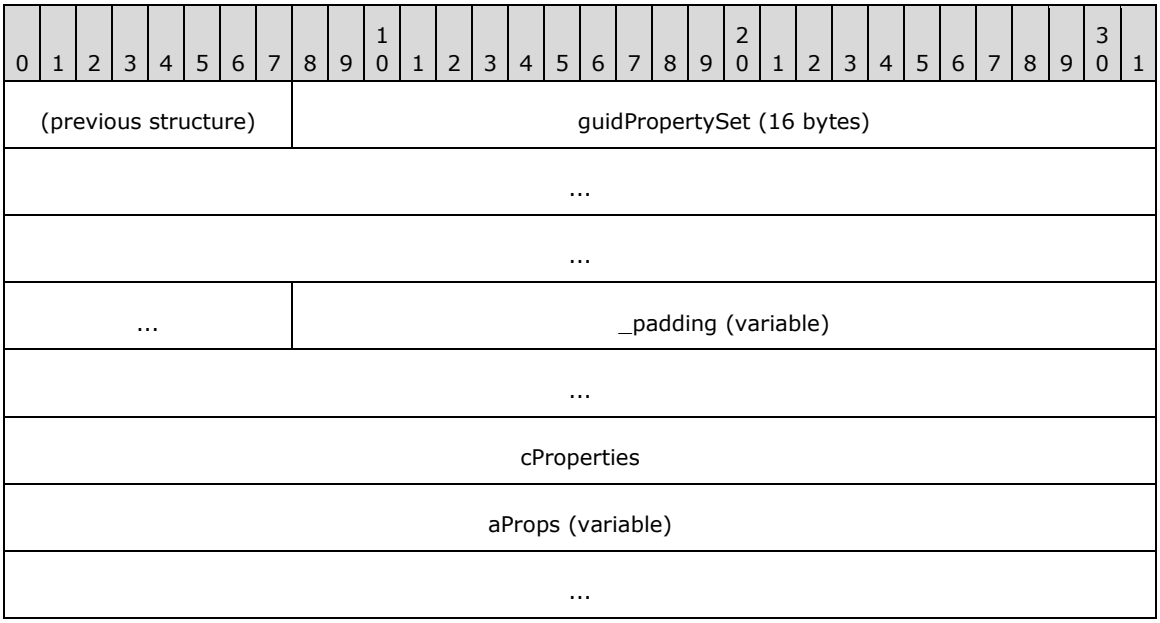
The following table lists the properties that are part of the DBPROPSET_CIFRMWRKCORE_EXT property set.

Value	Meaning
DBPROP_MACHINE 0x00000002	Specifies the names of the computers on which a query is to be processed. The value MUST be either VT_BSTR or VT_ARRAY | VT_BSTR.
DBPROP_CLIENT_CLSID	Specifies a connection constant for the GSS. The value MUST be a VT_CLSID

Value	Meaning
0x00000003	containing 0x2A4880706FD911D0A80800A0C906241A.

2.2.1.32 CDbPropSet

The CDbPropSet structure contains a set of properties. The first field (**guidPropertySet**) is not padded and will start where the previous structure in the message ended (as indicated by the "previous structure" entry in the diagram below). The 1-byte length of "previous structure" is arbitrary and is not meant to suggest that **guidPropertySet** will begin on any particular boundary. However, the **cProperties** field MUST be aligned to begin at a multiple of 4 bytes from the beginning of the message, and hence the format, is depicted as follows.



guidPropertySet (16 bytes): A GUID identifying the property set. MUST be set to the binary form corresponding to one of the following values (shown in string representation form), identifying the property set of the properties contained in the **aProps** field.

Value/GUID	Meaning
DBPROPSET_FSCIFRMWRK_EXT A9BD1526-6A80-11D0-8C9D-0020AF1D740E	File system content index framework property set.
DBPROPSET_QUERYEXT A7AC77ED-F8D7-11CE-A798-0020F8008025	Query extension property set.
DBPROPSET_CIFRMWRKCORE_EXT AFAFACA5-B5D1-11D0-8C62-00C04FC2DB8D	Content index framework core property set.

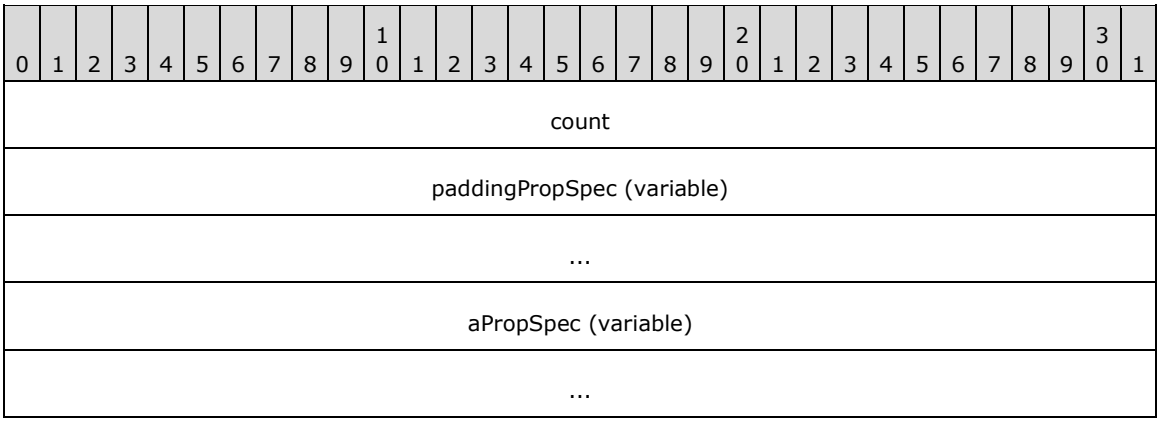
_padding (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver.

cProperties (4 bytes): A 32-bit unsigned integer containing the number of elements in the **aProps** array.

aProps (variable): An array of CDbProp structures containing properties. Structures in the array MUST be separated by 0 to 3 padding bytes such that each structure begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this array. If padding bytes are present, the value they contain is arbitrary. The content of the padding bytes MUST be ignored by the receiver.

2.2.1.33 CPidMapper

The CPidMapper structure contains an array of property specifications and serves to map from a property offset to a full property specification. The more compact property offsets are used to name properties in other parts of the protocol. Since offsets are more compact, they allow shorter property references in other parts of the protocol.



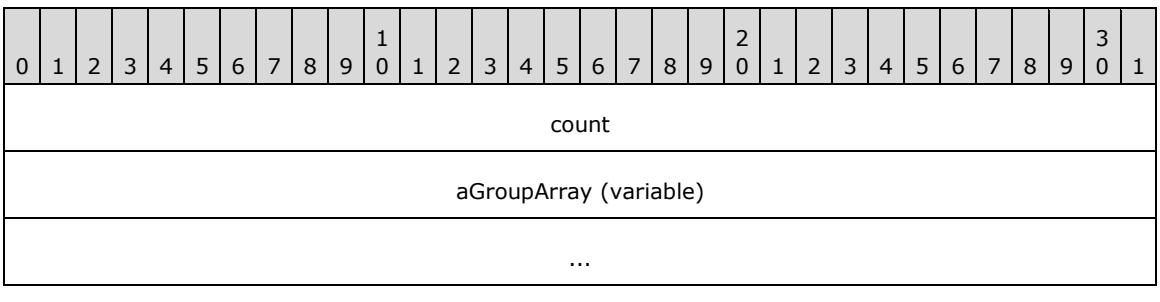
count (4 bytes): A 32-bit unsigned integer containing the number of elements in the **aPropSpec** array.

paddingPropSpec (variable): This field MUST be 0 to 4 bytes in length. The length of this field MUST be such that the byte offset from the beginning of the message to the first structure contained in the **aPropSpec** field is a multiple of 8. The value of the bytes can be any arbitrary value and MUST be ignored by the receiver.

aPropSpec (variable): Array of CFullPropSpec structures indicating the properties to return. Each CFullPropSpec in the array MUST be separated by 0 to 3 padding bytes such that each structure has a 4-byte alignment from the beginning of a message. Such padding bytes can be set to any arbitrary value when sent and MUST be ignored on receipt.

2.2.1.34 CColumnGroupArray

The CColumnGroupArray structure contains a set of property groups with weights for each property.

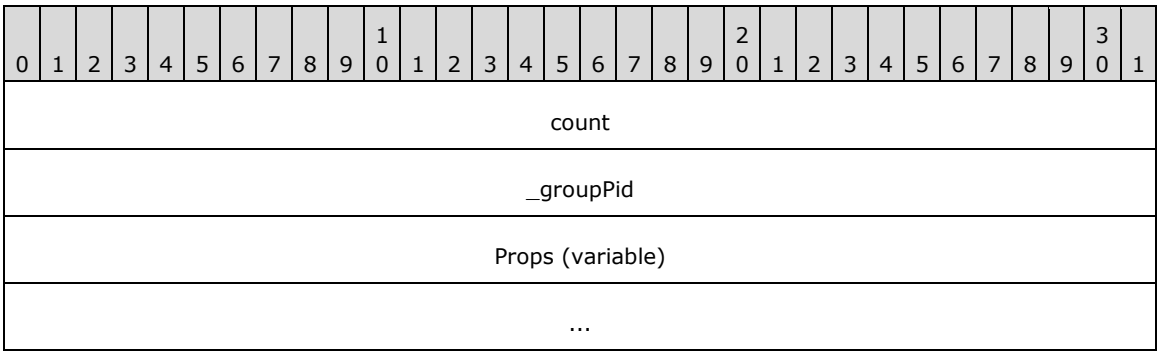


count (4 bytes): A 32-bit unsigned integer containing the number of elements in the **aGroupArray** array.

aGroupArray (variable): An array of CColumnGroup structures indicating individual weights for each property, which are used in probabilistic ranking. Structures in the array MUST be separated by 0 to 3 padding bytes such that each structure has a 4-byte alignment from the beginning of a message. Such padding bytes can be set to any arbitrary value when sent and MUST be ignored on receipt.

2.2.1.35 CColumnGroup

The CColumnGroup structure contains information about a property's weight in a single group.



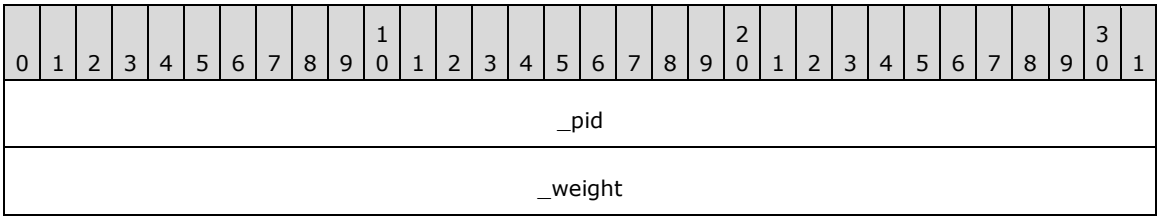
count (4 bytes): A 32-bit unsigned integer containing the number of elements in the **Props** array.

_groupPid (4 bytes): A 32-bit unsigned integer specifying group ID, a full property specification that can be used in the corresponding CProbRestriction. The value of **_groupPid** MUST satisfy the following result: (0xFFFF0000 & _groupPid) == 0x7FFF0000.

Props (variable): An array of SProperty structures, each specifying a PID and a weight for a property.

2.2.1.36 SProperty

The SProperty structure contains information about single property weight.

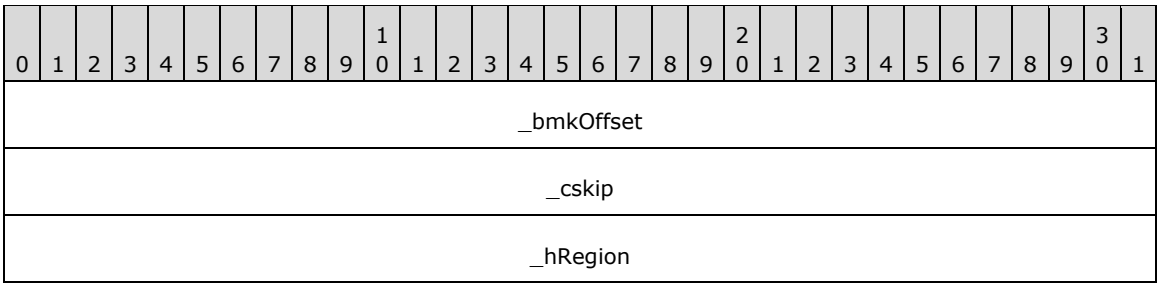


_pid (4 bytes): A 32-bit unsigned integer specifying a property identifier.

_weight (4 bytes): A 32-bit unsigned integer specifying the weight to be used in probabilistic ranking.

2.2.1.37 CRowSeekAt

The CRowSeekAt structure contains the offset at which to retrieve rows in a CPMGetRowsIn message.



_bmkOffset (4 bytes): A 32-bit value representing the handle of the bookmark indicating the starting position from which to skip the number of rows specified in **_cskip**, before beginning retrieval.

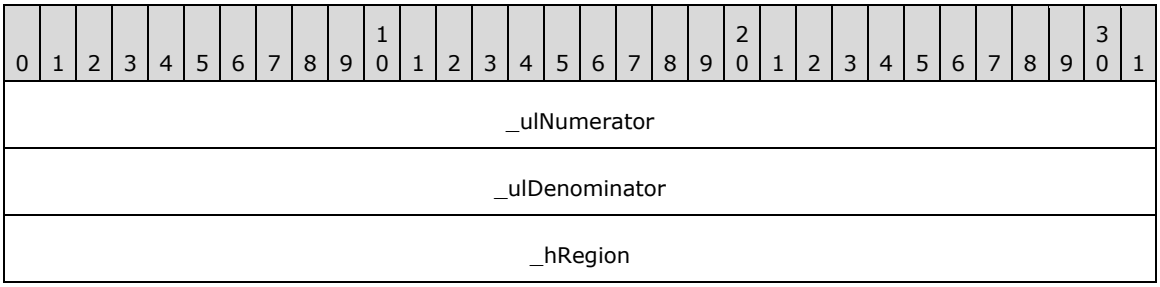
_cskip (4 bytes): A 32-bit unsigned integer containing the number of rows to skip in the rowset.

_hRegion (4 bytes): A 32-bit unsigned integer.

Note This field MUST be set to 0x00000000 and MUST be ignored.

2.2.1.38 CRowSeekAtRatio

The CRowSeekAtRatio structure identifies the point at which to begin retrieval for a CPMGetRowsIn message.



_ulNumerator (4 bytes): A 32-bit unsigned integer representing the numerator of the ratio of rows in the chapter at which to begin retrieval.

_ulDenominator (4 bytes): A 32-bit unsigned integer representing the denominator of the ratio of rows in the chapter at which to begin retrieval. This MUST be greater than zero.

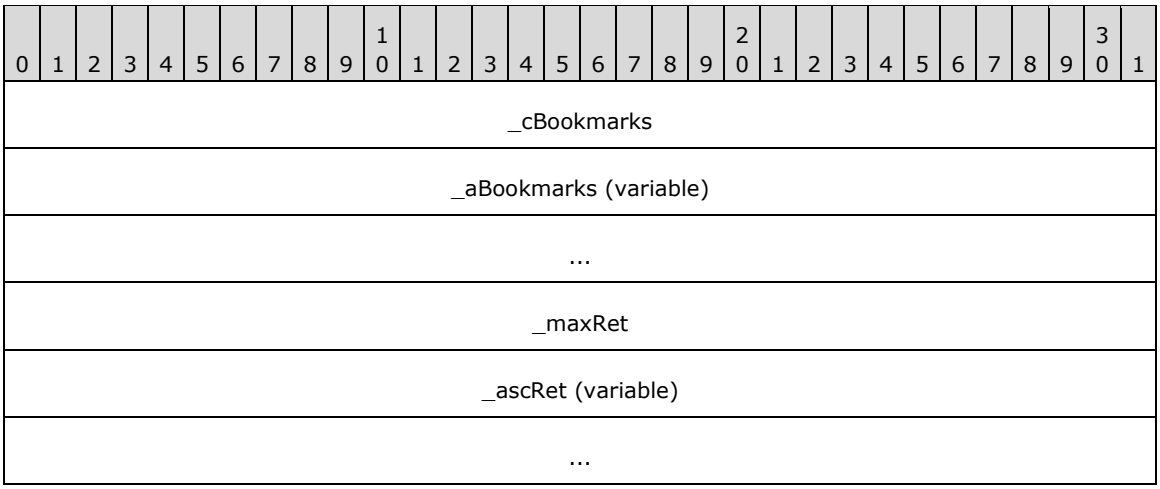
_hRegion (4 bytes): A 32-bit unsigned integer.

Note This field MUST be set to 0x00000000 and MUST be ignored.

2.2.1.39 CRowSeekByBookmark

The CRowSeekByBookmark structure identifies the bookmarks from which to begin retrieving rows for a CPMGetRowsIn message.

Note The bookmark handles are first retrieved with a CPMGetRowsOut message. Once retrieved, they can be used, as part of the CRowSeekByBookmark structure, to retrieve the bookmarked data using a CPMGetRowsIn message.



_cBookmarks (4 bytes): A 32-bit unsigned integer representing the number of elements in _aBookmarks array.

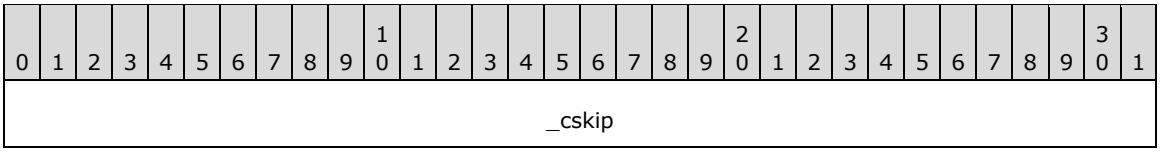
_aBookmarks (variable): An array of bookmark handles (each represented by 4 bytes), as obtained from a previous CPMGetRowsOut message.

_maxRet (4 bytes): A 32-bit unsigned integer representing the number of elements in the **_ascRet** array.

_ascRet (variable): An array of HRESULT values. When the CRowSeekByBookmark is sent as part of the CPMGetRowsIn request, the number of entries in the array MUST be equal to **_maxRet**. When sent by the client, MUST be set to zero when sent and MUST be ignored on receipt. When sent by the server (as part of the CPMGetRowsOut message), the values in the array indicate the result status for each row retrieval.

2.2.1.40 CRowSeekNext

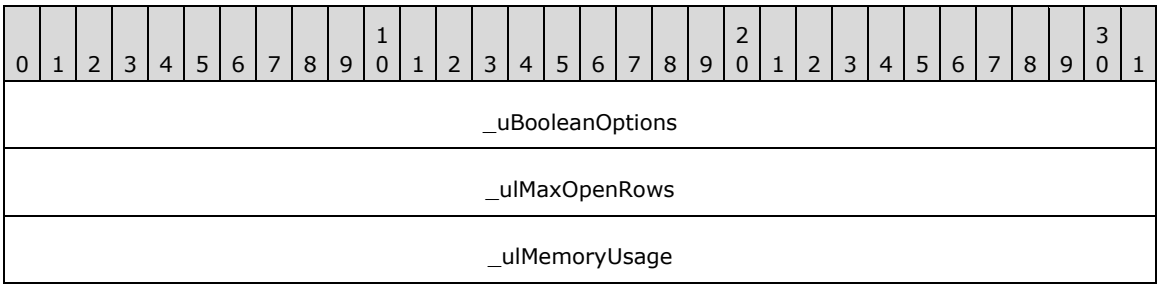
The CRowSeekNext structure contains the number of rows to skip in a CPMGetRowsIn message.



_cskip (4 bytes): A 32-bit unsigned integer representing the number of rows to skip in the rowset.

2.2.1.41 CRowsetProperties

The CRowsetProperties structure contains configuration information for a query.



_cMaxResults
_cCmdTimeout

_uBooleanOptions (4 bytes): The least significant 3 bits of this field MUST contain one of the following three values.

Value	Meaning
eSequential 0x00000001	The cursor can only be moved forward.
eLocatable 0x00000003	The cursor can be moved to any position.
eScrollable 0x00000007	The cursor can be moved to any position and fetch in any direction.

The remaining bits can be clear or set to any combination of the following values logically ORed together.

Value	Meaning
eAsynchronous 0x00000008	The client will not wait for execution completion.
eFirstRows 0x00000080	Return the first rows encountered, not the best matches.
eHoldRows 0x00000200	The server MUST NOT discard rows until the client is done with a query.
eChaptered 0x00000800	The rowset supports chapters.
eUseCI 0x00001000	Use the inverted index to evaluate content restrictions even if it is out of date. If not set, the GSS can opt to execute the query by going directly against the file system.
eDeferTrimming 0x00002000	Non-indexed trimming operations like scoping or security checking can be expensive. This option gives the GSS the option of deferring these operations until rows are actually requested.
eEnableRowsetEvents 0x00800000	Enables storage of rowset events on the server side. (For information about how to retrieve stored events, see the CPMGetRowsetNotifyIn message.)
eDoNotComputeExpensiveProps 0x00400000	Prevents computation of expensive properties. Windows implementations treat **cRowsTotal**, **_maxRank**, and **_cResultsFound** (as specified in CPMGetQueryStatusExOut (section 2.2.3.9)) as expensive properties. Other implementations could choose different properties and mark them as expensive.

_ulMaxOpenRows (4 bytes): A 32-bit unsigned integer.

Note This field MUST be set to 0x00000000. It is not used and MUST be ignored.

_ulMemoryUsage (4 bytes): A 32-bit unsigned integer.

Note This field MUST be set to 0x00000000. It is not used and MUST be ignored.

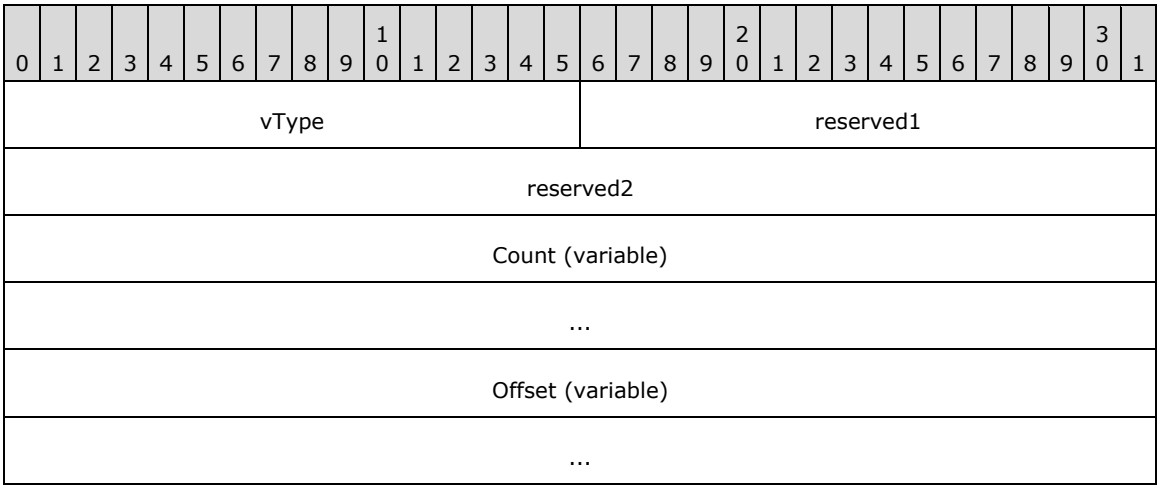
_cMaxResults (4 bytes): A 32-bit unsigned integer specifying the maximum number of rows that are to be returned for the query.

_cCmdTimeout (4 bytes): A 32-bit unsigned integer, specifying the number of seconds at which a query is to time out and automatically terminate, counting from the time the query starts executing on the server.

Note A value of 0x00000000 means that the query is not to time out.

2.2.1.42 CTableVariant

The CTableVariant structure contains the fixed-size portion of a variable length data type stored in the CPMGetRowsOut message.



vType (2 bytes): A type indicator, indicating the type of **vValue**. It MUST be one of the values under the **vType** field, as specified in section 2.2.1.1.

reserved1 (2 bytes): Not used. Can be set to any arbitrary value when sent and it MUST be ignored on receipt.

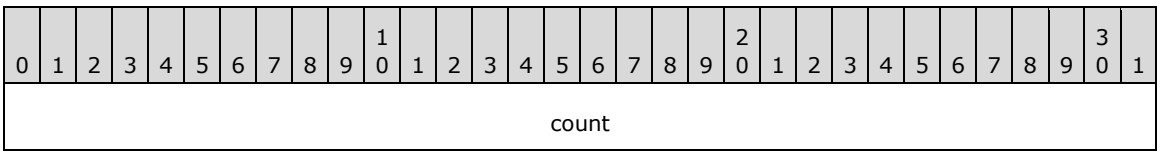
reserved2 (4 bytes): Not used. Can be set to any arbitrary value when sent and it MUST be ignored on receipt.

Count (variable): Element count. This field is 4 bytes and is present only if the **vType** is a VT_VECTOR.

Offset (variable): An offset to variable length data (for example, a string). This MUST be a 32-bit value (4 bytes long) if 32-bit offsets are being used (per the rules in section 2.2.3.12), or a 64-byte value (8 bytes long) if 64-bit offsets are being used.

2.2.1.43 CSortSet

The CSortSet structure contains the sort order of the query.



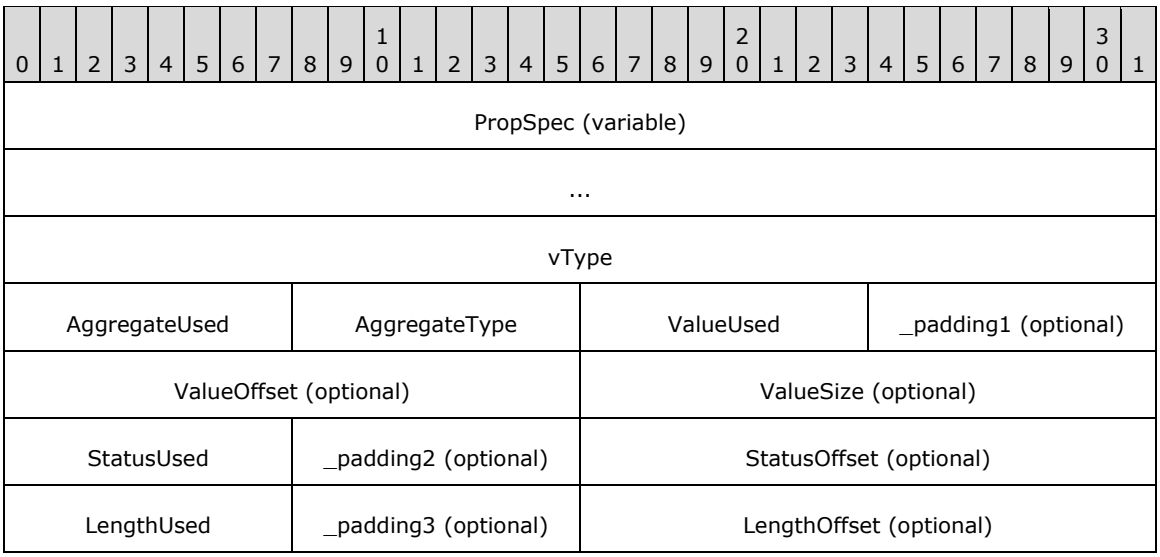
sortArray (variable)
...

count (4 bytes): A 32-bit unsigned integer specifying the number of elements in **sortArray**.

sortArray (variable): An array of CSort structures describing the order in which to sort the results of the query. Structures in the array MUST be separated by 0 to 3 padding bytes such that each structure has a 4-byte alignment from the beginning of a message. Such padding bytes can be set to any arbitrary value when sent and MUST be ignored on receipt.

2.2.1.44 CTableColumn

The CTableColumn structure contains a column of a CPMSetBindingsIn message.



PropSpec (variable): A CFullPropSpec structure.

vType (4 bytes): A 32-bit unsigned integer that specifies the type of data value contained in the column. See the **vType** field in section 2.2.1.1 for the list of values for this field.

AggregateUsed (1 byte): MUST be set to one of the following values.

Value	Meaning
0x00	No aggregation is used, and the **AggregateType** field MUST NOT be present.
0x01	This column is used to aggregate the values for query results, as specified in **AggregateType**.

AggregateType (1 byte): This field MUST be set to one of the aggregation type values specified under the **type** field in section 2.2.1.25.

ValueUsed (1 byte): A 1-byte field that MUST be set to one of the following values.

Value	Meaning
0x00	The value of the column is not transferred in the row.

Value	Meaning
0x01	The value of the column is transferred within the row.

_padding1 (1 byte): A padding field.

Note This field MUST be inserted before **ValueOffset** if, without it, **ValueOffset** would not begin at an even offset from the beginning of the message. The value of this byte is arbitrary and MUST be ignored. If **ValueUsed** is set to 0x00, this field MUST NOT be present.

ValueOffset (2 bytes): An unsigned 2-byte integer specifying the offset of the column value in the row. If **ValueUsed** is set to 0x00, this field MUST NOT be present.

ValueSize (2 bytes): An unsigned 2-byte integer specifying the size of the column value in bytes. If **ValueUsed** is set to 0x00, this field MUST NOT be present.

StatusUsed (1 byte): MUST be set to one of the following values.

Value	Meaning
0x00	The status of the column is not transferred within the row.
0x01	The status of the column is transferred within the row.

_padding2 (1 byte): A padding field.

Note This field MUST be inserted before **StatusOffset** if, without it, the **StatusOffset** field would not begin at an even offset from the beginning of the message. The value of this byte is arbitrary and MUST be ignored. If **StatusUsed** is set to 0x00, this field MUST NOT be present.

StatusOffset (2 bytes): An unsigned 2-byte integer.

Note Specifies the offset of the column status in the row. If **StatusUsed** is set to 0x00, this field MUST NOT be present.

LengthUsed (1 byte): A 1-byte field that MUST be set to one of the following values.

Value	Meaning
0x00	The length of the column MUST NOT be transferred within the row.
0x01	The length of the column is transferred within the row.

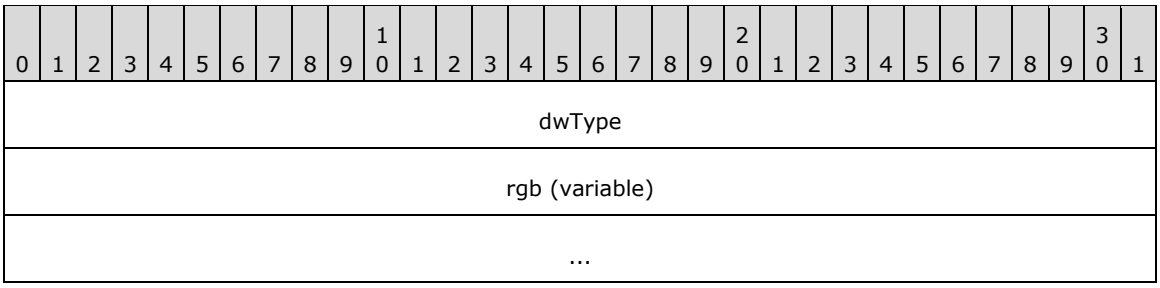
_padding3 (1 byte): A padding field.

Note This field MUST be inserted before **LengthOffset** if, without it, **LengthOffset** would not begin at an even offset from the beginning of a message. The value of this byte is arbitrary and MUST be ignored. If **LengthUsed** is set to 0x00, this field MUST NOT be present.

LengthOffset (2 bytes): An unsigned 2-byte integer specifying the offset of the column length in the row. In CPMGetRowsOut, length is represented by a 32-bit unsigned integer by the offset specified in **LengthOffset**. If **LengthUsed** is set to 0x00, this field MUST NOT be present.

2.2.1.45 SERIALIZEDPROPERTYVALUE

The SERIALIZEDPROPERTYVALUE structure contains a serialized value.

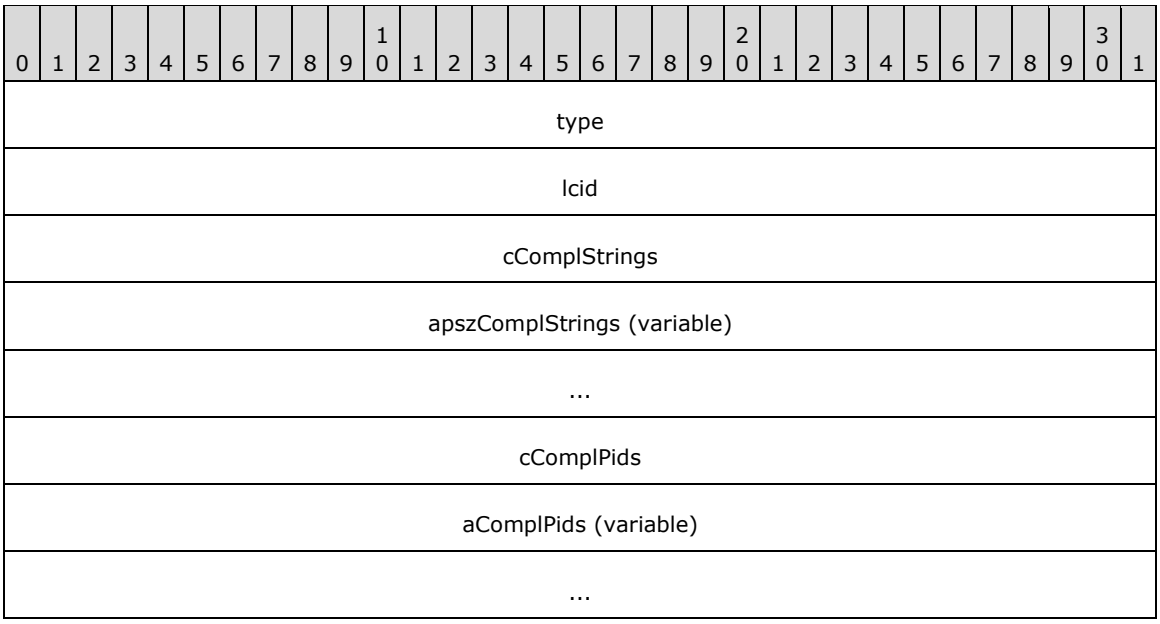


dwType (4 bytes): One of the variant types, as defined in section 2.2.1.1, that can be combined with variant type modifiers. For all variant types, except those combined with VT_ARRAY, SERIALIZEDPROPERTYVALUE has the same layout as CBaseStorageVariant. If the variant type is combined with the VT_ARRAY type modifier, SAFEARRAY2 is used instead of SAFEARRAY in the **vValue** field of CBaseStorageVariant.

rgb (variable): Serialized value. The serialization depends on the value of **dwType**, which is used identically to the **vValue** field in section 2.2.1.1. The process used to serialize **rgb** is the same as that described in section 2.2.1.1.

2.2.1.46 CCompletionCategSpec

The CCompletionCategSpec structure contains the specification for building groups of search completion suggestions.



type (4 bytes): A 32-bit unsigned integer. 0 for fuzzy matching, 1 for exact. A match is exact if it is a prefix of any of the strings specified by **apszComplStrings**. Fuzzy matching specifies that the server return results that are similar to the primary query string for a server-defined definition of "similar".

lcid (4 bytes): A 32-bit unsigned integer, indicating the locale ID of the query strings.

cComplStrings (4 bytes): A 32-bit unsigned integer, indicating how many query strings will follow.

apszComplStrings (variable): An array of SERIALIZEDPROPERTYVALUE structures. Each structure represents a single query string. The first is the primary query string, and it is the structure that

will be fuzzy-matched if the type value indicates fuzzy matching. The rest of the query strings are always matched by exact prefix. There will be a total of **cComplStrings** values.

cComplPids (4 bytes): A 32-bit unsigned integer, indicating how many PIDs will follow.

aComplPids (variable): An array of 32-bit unsigned integers. Each one is a PID representing a property that can be matched against for search completion values. If no PIDs are provided, all PIDs that are included in the index of completion strings are considered for matches. There will be **cComplPids** PIDs.

2.2.2 Message Headers

All Windows Search Protocol messages have a 16-byte header.

The following table shows the Windows Search Protocol message header format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_msg																															
_status																															
_ulChecksum																															
_ulReserved2																															

_msg (4 bytes): A 32-bit integer that identifies the type of message following the header. The following table lists the Windows Search Protocol messages and the integer values specified for each message. As shown in the table, some values identify two messages. In those instances, the message following the header can be identified by the direction of the message flow. If the direction is client to server, the message with "In" appended to the message name is indicated. If the direction is server to client, the message with "Out" appended to the message name is indicated.

Value	Meaning
0x000000C8	CPMConnectIn or CPMConnectOut
0x000000C9	CPMDisconnect
0x000000CA	CPMCreateQueryIn or CPMCreateQueryOut
0x000000CB	CPMFreeCursorIn or CPMFreeCursorOut
0x000000CC	CPMGetRowsIn or CPMGetRowsOut
0x000000CD	CPMRatioFinishedIn or CPMRatioFinishedOut
0x000000CE	CPMCompareBmkIn or CPMCompareBmkOut
0x000000CF	CPMGetApproximatePositionIn or CPMGetApproximatePositionOut
0x000000D0	CPMSetBindingsIn
0x000000D1	CPMGetNotify
0x000000D2	CPMSendNotifyOut

Value	Meaning
0x000000D7	CPMGetQueryStatusIn or CPMGetQueryStatusOut
0x000000D9	CPMCiStateInOut
0x000000E4	CPMFetchValueIn or CPMFetchValueOut
0x000000E7	CPMGetQueryStatusExIn or CPMGetQueryStatusExOut
0x000000E8	CPMRestartPositionIn
0x000000EC	CPMSetCatStateIn (not supported)
0x000000F1	CPMGetRowsetNotifyIn or CPMGetRowsetNotifyOut
0x000000F2	CPMFindIndicesIn, or CPMFindIndicesOut
0x000000F3	CPMSetScopePrioritizationIn or CPMSetScopePrioritizationOut
0x000000F4	CPMGetScopeStatisticsIn or CPMGetScopeStatisticsOut

_status (4 bytes): An HRESULT, indicating the status of the requested operation. The client MUST initialize this value to 0x00000000. The server then changes it as the status of the requested operation changes.

_ulChecksum (4 bytes): The _ulChecksum MUST be calculated as specified in section 3.2.4 for the following messages:

- CPMConnectIn
- CPMCreateQueryIn
- CPMSetBindingsIn
- CPMGetRowsIn
- CPMFetchValueIn

Note For all other messages, _ulChecksum MUST be set to 0x00000000. A client MUST ignore the _ulChecksum field.

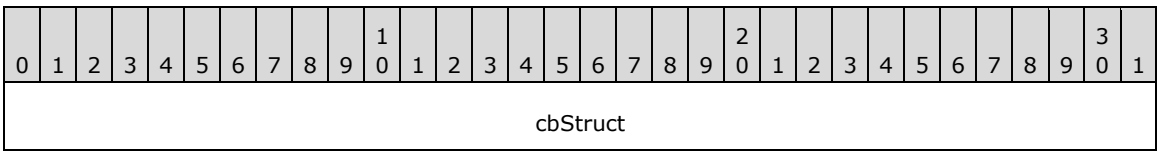
_ulReserved2 (4 bytes): MUST be ignored by the receiver.

Note This field MUST be set to 0x00000000 except for the CPMGetRowsIn message, where it MUST hold the high 32-bits part of a 64-bit offset if 64-bit offsets are being used (see section 2.2.3.12 for details).

2.2.3 Messages

2.2.3.1 CPMCiStateInOut

The CPMCiStateInOut message contains information about the state of the GSS.



cWordList
cPersistentIndex
cQueries
cDocuments
cFreshTest
dwMergeProgress
eState
cFilteredDocuments
cTotalDocuments
cPendingScans
dwIndexSize
cUniqueKeys
cSecQDocuments
dwPropCacheSize

cbStruct (4 bytes): A 32-bit unsigned integer indicating the size, in bytes, of this message (excluding the common header). MUST be set to 0x0000003C.

cWordList (4 bytes): A 32-bit unsigned integer containing the number of in-memory indexes created for recently indexed documents.

cPersistentIndex (4 bytes): A 32-bit unsigned integer containing the number of persisted indexes.

cQueries (4 bytes): A 32-bit unsigned integer indicating a number of actively running queries.

cDocuments (4 bytes): A 32-bit unsigned integer indicating the total number of documents waiting to be indexed.

cFreshTest (4 bytes): A 32-bit unsigned integer indicating the number of unique documents with information in indexes that are not fully optimized for performance.

dwMergeProgress (4 bytes): A 32-bit unsigned integer specifying the completion percentage of current full optimization of indexes while optimization is in progress. MUST be less than or equal to 100.

eState (4 bytes): A 32-bit unsigned integer indicating the state of content indexing. MUST be zero or one or more of the CI_STATE_* constants defined in the following table.

Value	Meaning
0x00000000	None of the following states apply.

Value	Meaning
CI_STATE_SHADOW_MERGE 0x00000001	The GSS is in the process of optimizing some of the indexes to reduce memory usage and improve query performance.
CI_STATE_MASTER_MERGE 0x00000002	The GSS is in the process of full optimization for all indexes.
CI_STATE_CONTENT_SCAN_REQUIRED 0x00000004	Some documents in the inverted index have changed and the GSS needs to determine which have been added, changed, or deleted.
CI_STATE_ANNEALING_MERGE 0x00000008	The GSS is in the process of optimizing indexes to reduce memory usage and improve query performance. This process is more comprehensive than the one identified by the CI_STATE_SHADOW_MERGE value, but it is not as comprehensive as specified by the CI_STATE_MASTER_MERGE value. Such optimizations are implementation-specific as they depend on the way data is stored internally; the optimizations do not affect the protocol in any way other than response time.
CI_STATE_SCANNING 0x00000010	The GSS is checking a directory or a set of directories to determine if any files have been added, deleted, or updated since the last time the directory was indexed.
CI_STATE_LOW_MEMORY 0x00000080	Most of the virtual memory of the server is in use.
CI_STATE_HIGH_IO 0x00000100	The level of input/output (I/O) activity on the server is relatively high.
CI_STATE_MASTER_MERGE_PAUSED 0x00000200	The process of full optimization for all indexes that was in progress has been paused. This is given for informative purposes only and does not affect the Windows Search Protocol.
CI_STATE_READ_ONLY 0x00000400	The portion of the GSS that picks up new documents to index has been paused. This is given for informative purposes only and does not affect the Windows Search Protocol.
CI_STATE_BATTERY_POWER 0x00000800	The portion of the GSS that picks up new documents to index has been paused to conserve battery lifetime but still replies to the queries. This is given for informative purposes only and does not affect the Windows Search Protocol.
CI_STATE_USER_ACTIVE 0x00001000	The portion of the GSS that picks up new documents to index has been paused due to high activity by the user (keyboard or mouse) but still replies to the queries. This is given for informative purposes only and does not affect the Windows Search Protocol.
CI_STATE_LOW_DISK 0x00010000	The service is paused due to low disk availability.
CI_STATE_HIGH_CPU 0x00020000	The service is paused due to high CPU usage.

cFilteredDocuments (4 bytes): A 32-bit unsigned integer indicating the number of documents indexed since content indexing began.

cTotalDocuments (4 bytes): A 32-bit unsigned integer indicating the total number of documents in the system.

cPendingScans (4 bytes): A 32-bit unsigned integer indicating the number of pending high-level indexing operations. The meaning of this value is provider-specific, but larger numbers are expected to indicate that more indexing remains.<4>

dwIndexSize (4 bytes): A 32-bit unsigned integer indicating the size, in megabytes, of the index (excluding the property cache).

cUniqueKeys (4 bytes): A 32-bit unsigned integer indicating the approximate number of unique keys in the catalog.

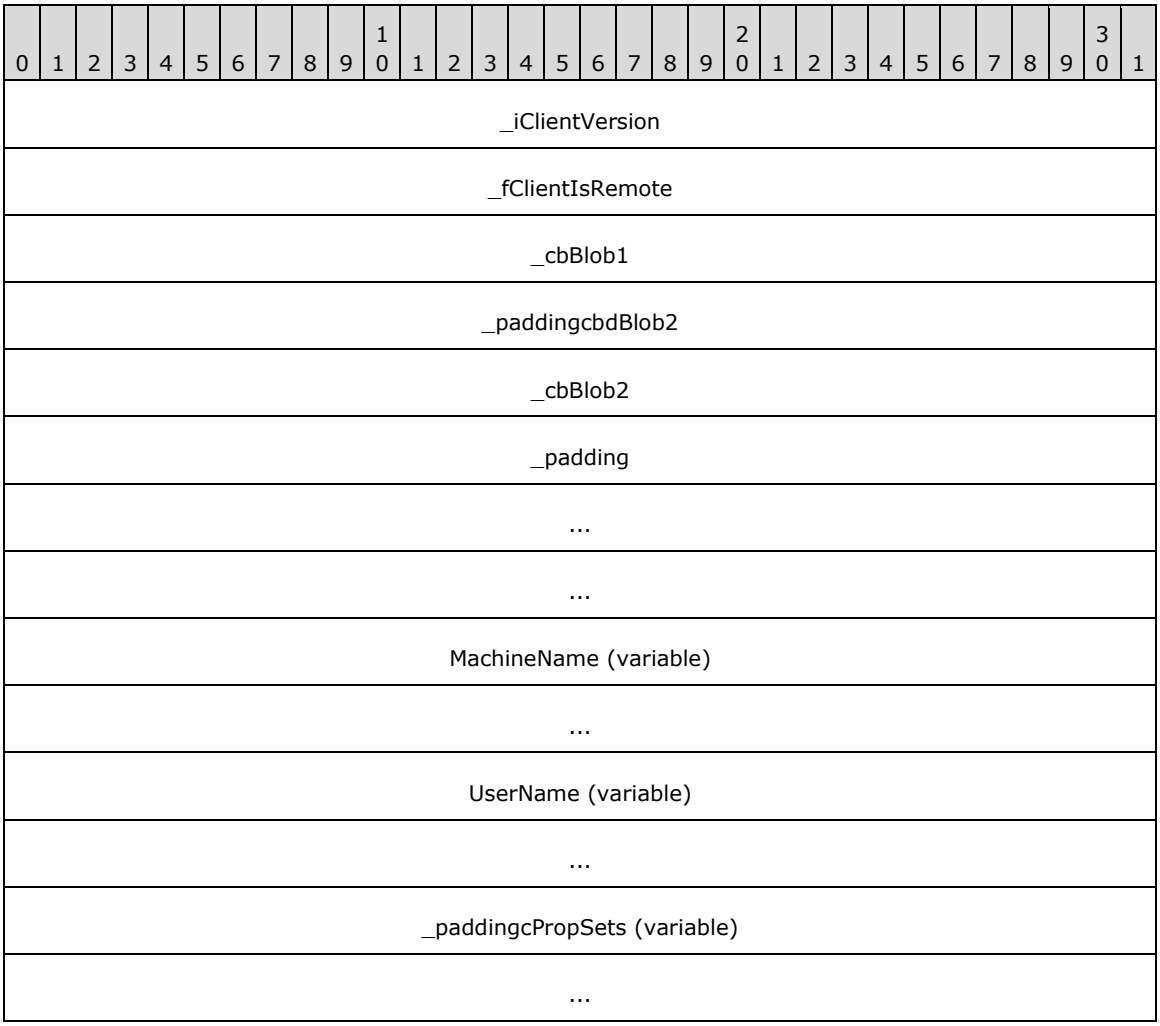
cSecQDocuments (4 bytes): A 32-bit unsigned integer indicating the number of documents that the GSS will attempt to index again because of a failure during the initial indexing attempt.

dwPropCacheSize (4 bytes): A 32-bit unsigned integer indicating the size, in megabytes, of the property cache.

2.2.3.2 CPMConnectIn

The CPMConnectIn message begins a session between the client and server.

The format of the CPMConnectIn message that follows the header is shown in the following diagram.



cPropSets
PropertySet1 (variable)
...
PropertySet2 (variable)
...
PaddingExtPropset (variable)
...
cExtPropSet
aPropertySets (variable)
...

_iClientVersion (4 bytes): A 32-bit integer indicating whether the server is to validate the checksum value specified in the **_ulChecksum** field of the message headers for messages sent by the client.

Note If the **_iClientVersion** field's lowest 2 bytes are set to 0x00000109 or greater, the server MUST validate the **_ulChecksum** field value for the following messages:

- CPMConnectIn
- CPMCreateQueryIn
- CPMFetchValueIn
- CPMGetRowsIn
- CPMSetBindingsIn

For details on how the server validates the value specified by the client in the **_ulChecksum** field for the messages previously listed, see section 3.2.4.

If the lowest 2 bytes are greater than 0x00000109, the client is assumed to be capable of handling 64-bit offsets in CPMGetRowsOut messages.<5><6>

_fClientIsRemote (4 bytes): A Boolean value indicating if the client is running on a different machine than the server. This field is set to 0x00000001 if the client is running on a different machine and 0x00000000 if it is not.

_cbBlob1 (4 bytes): A 32-bit unsigned integer indicating the size, in bytes, of the **cPropSets**, **PropertySet1**, and **PropertySet2** fields combined.

_paddingcbdBlob2 (4 bytes): This field MUST be 4 bytes in length. The length of this field MUST be such that the byte offset from the beginning of the message to the beginning of the **_cbBlob2** field is a multiple of 8. The value of the bytes can be any arbitrary value, and MUST be ignored by the receiver.

_cbBlob2 (4 bytes): A 32-bit unsigned integer indicating the size in bytes of the **cExtPropSet** and **aPropertySet** fields, combined.

_padding (12 bytes): Twelve bytes of padding that can contain arbitrary values and MUST be ignored.

MachineName (variable): The machine name of the client. The name string MUST be a null-terminated array of less than 512 Unicode characters, including the null terminator. The server MUST ignore this field upon receipt.

UserName (variable): A string that represents the user name of the person who is running the application that invoked this protocol. The name string MUST be a null-terminated array of less than 512 Unicode characters when concatenated with **MachineName**. The server MUST ignore this field upon receipt.

_paddingcPropSets (variable): This field MUST be 0 to 7 bytes in length. The number of bytes MUST be the number required to make the byte offset of the **cPropSets** field from the beginning of the message that contains this structure equal a multiple of 8. The value of the bytes can be any arbitrary value, and MUST be ignored by the receiver.

cPropSets (4 bytes): A 32-bit unsigned integer indicating the number of CDbPropSet structures following this field.

Note This field MUST be set to 0x0000002.

PropertySet1 (variable): A CDbPropSet structure with **guidPropertySet** containing DBPROPSET_FSCIFRMWRK_EXT.

PropertySet2 (variable): A CDbPropSet structure with **guidPropertySet** containing DBPROPSET_CIFRMWRKCORE_EXT.

PaddingExtPropset (variable): This field MUST be 0 to 7 bytes in length. The number of bytes MUST be the number required to make the byte offset of the **cExtPropSets** field from the beginning of the message that contains this structure equal a multiple of 8. The value of the bytes can be any arbitrary value, and MUST be ignored by the receiver.

cExtPropSet (4 bytes): A 32-bit unsigned integer indicating the number of CDbPropSet structures following this field. This field must be greater than or equal to 1.

aPropertySets (variable): An array of CDbPropSet structures specifying other properties. The number of elements in this array MUST be equal to **cExtPropSet**.

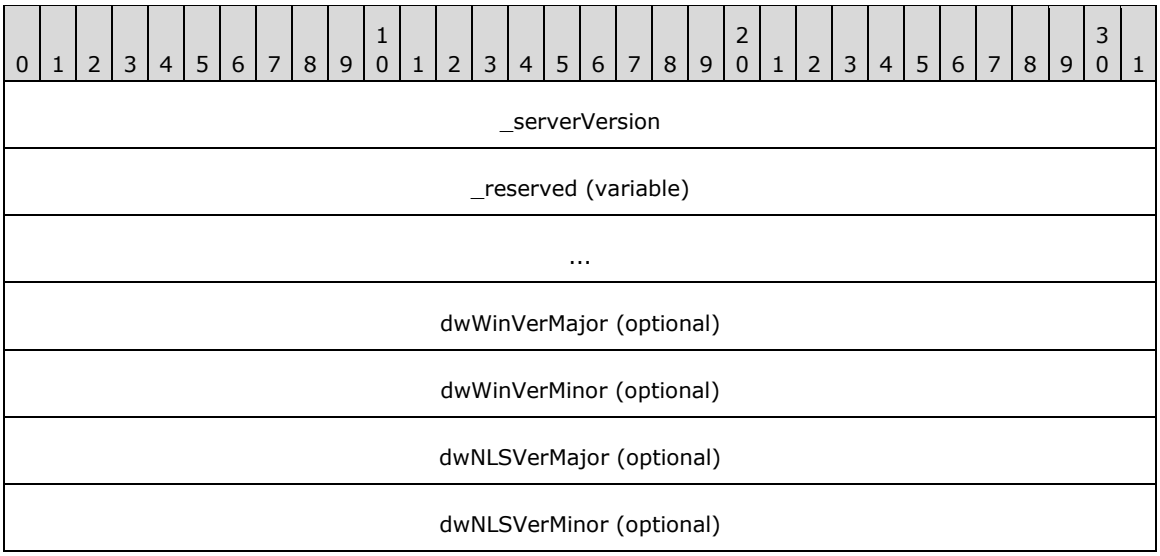
Although any value can be passed here, the server MUST ignore anything except the following property, which must be the first property in **aPropertySets**. This value must also align with the DBPROP_CI_CATALOG_NAME specified in DBPROPSET_FSCIFRMWRK_EXT.

Value	Meaning
DBPROP_CI_CATALOG_NAME 0x00000002	This is the catalog name for the query.

2.2.3.3 CPMConnectOut

The CPMConnectOut message contains a response to a CPMConnectIn message.

The format of the CPMConnectOut message that follows the header is shown in the following diagram.



_serverVersion (4 bytes): A 32-bit integer that indicates whether the server can support 64-bit offsets. Values greater than or equal to 0x00010000 indicate 64-bit support. Values less than 0x00010000 indicate 32-bit support.<7><8>

_reserved (variable): The server can send an arbitrary number of arbitrary values, and the client MUST ignore these values if present. If the server supports version reporting, the size MUST be 4 bytes.

dwWinVerMajor (4 bytes): 32-bit unsigned integer that contains the major version number of the Windows operating system on a server. If server doesn't supports version reporting then this field MUST be omitted.<9>

If present this field can contain one of the following values:

Value	Meaning
WINDOWS_MAJOR_VERSION_6 0x00000006	The major version of the Windows operating system is 0x00000006.

dwWinVerMinor (4 bytes): 32-bit unsigned integer that contains the minor version number of the Windows operating system on a server. If server doesn't supports version reporting then this field MUST be omitted. <10>

If present this field can contain one of the following values:

Value	Meaning
WINDOWS_MINOR_VERSION_0 0x00000000	The minor version of the Windows operating system is 0x00000000.
WINDOWS_MINOR_VERSION_1 0x00000001	The minor version of the Windows operating system is 0x00000001.

dwNLSVerMajor (4 bytes): 32-bit unsigned integer that contains the National Language Support (NLS) version number of the Windows operating system on a server. If server doesn't supports version reporting then this field MUST be omitted. <11>

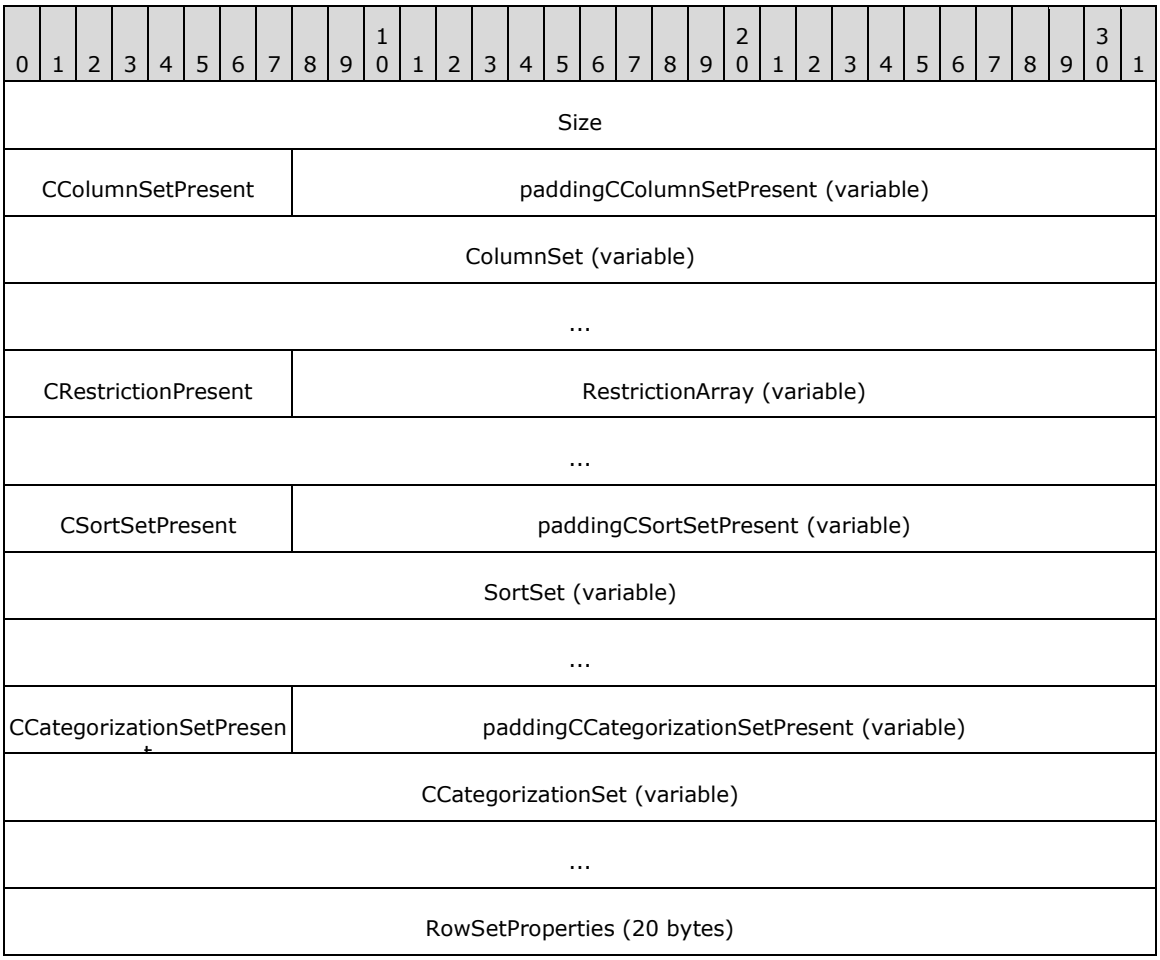
dwNLSVerMinor (4 bytes): 32-bit unsigned integer that contains the defined National Language Support (NLS) version number of the Windows operating system on a server. If server doesn't supports version reporting then this field MUST be omitted. <12>

If present, the **dwNLSVerMajor** and **dwNLSVerMinor** fields can contain one of the following values:

Value	Meaning
NLS_ VERSION_40500 0x00040500	Defined NLS version is 0x00040500.
NLS_VERSION_60000 0x00060000	Defined NLS version is 0x00060000.
NLS_VERSION_60101 0x00060101	Defined NLS version is 0x00060101.

2.2.3.4 CPMCreateQueryIn

The CPMCreateQueryIn message creates a new query. The format of the CPMCreateQueryIn message that follows the header is shown in the following diagram.



...
...
PidMapper (variable)
...
GroupArray (variable)
...
Lcid

Size (4 bytes): A 32-bit unsigned integer indicating the number of bytes from the beginning of this field to the end of the message.

CColumnSetPresent (1 byte): A byte field indicating if the **ColumnSet** field is present. MUST be set to one of the following values. If the value is set to 0x00, then no information will be returned from the query.

Value	Meaning
0x00	The **ColumnSet** field MUST be absent.
0x01	The **ColumnSet** field MUST be present.

paddingCColumnSetPresent (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver. This field MUST be absent in CColumnSetPresent is set to 0x00.

ColumnSet (variable): A CColumnSet structure containing the property offsets for properties in CPidMapper that are returned as a column. If no properties are in the column set, then no information will be returned from the query.

CRestrictionPresent (1 byte): A byte field indicating whether the **RestrictionArray** field is present.

Note If set to any nonzero value, the **RestrictionArray** field MUST be present. If set to 0x00, **RestrictionArray** MUST be absent.

RestrictionArray (variable): A CRestrictionArray structure containing the command tree of the query.

CSortSetPresent (1 byte): A byte field indicating whether the **SortSet** field is present.

Note If set to any nonzero value, the **SortSet** field MUST be present. If set to 0x00, **SortSet** MUST be absent.

paddingCSortSetPresent (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver. This field MUST be absent if CSortSetPresent is set to 0x00.

SortSet (variable): A CInGroupSortAggregSets structure indicating the sort order of the query.

CCategorizationSetPresent (1 byte): A byte field indicating whether the **CCategorizationSet** field is present.

Note If set to any nonzero value, the **CCategorizationSet** field MUST be present. If set to 0x00, **CCategorizationSet** MUST be absent.

paddingCCategorizationSetPresent (variable): This field MUST be 0 to 3 bytes in length. The length of this field MUST be such that the following field begins at an offset that is a multiple of 4 bytes from the beginning of the message that contains this structure. If this field is present (that is, length nonzero), the value it contains is arbitrary. The content of this field MUST be ignored by the receiver. This field MUST be absent if CCategorizationSetPresent is set to 0x00.

CCategorizationSet (variable): A CCategorizationSet structure that contains the groups for the query.

RowSetProperties (20 bytes): A CRowsetProperties structure providing configuration information for the query.

PidMapper (variable): A CPidMapper structure that maps from property offsets to full property descriptions.

GroupArray (variable): A CColumnGroupArray structure, describing property weights for probabilistic ranking.

Lcid (4 bytes): A 32-bit unsigned integer, indicating the user's locale for this query, as specified in [MS-LCID].

2.2.3.5 CPMCreateQueryOut

The CPMCreateQueryOut message contains a response to a CPMCreateQueryIn message.

The format of the CPMCreateQueryOut message that follows the header is shown in the following table.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_fTrueSequential																															
_fWorkIdUnique																															
aCursors																															

_fTrueSequential (4 bytes): A 32-bit unsigned integer. MUST be set to one of the following values.

Note An informative value indicating whether the query can be expected to provide results faster.

Value	Meaning
0x00000000	For the query provided in CPMCreateQueryIn, there would be a greater latency in delivering query results.
0x00000001	If **_fTrueSequential** is set to true, results can be returned sequentially without the server incurring the cost of processing the entire result set before returning the first result.

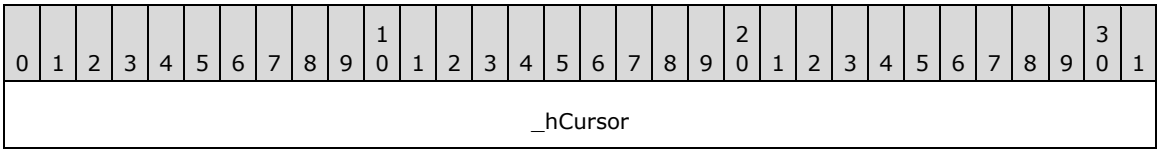
_fWorkIdUnique (4 bytes): A Boolean value indicating whether the document identifiers pointed by the cursors are unique throughout query results. MUST be set to one of the following values.

Value	Meaning
0x00000000	The cursors are unique only throughout the rowset.
0x00000001	The cursors are unique across multiple query results.

aCursors (4 bytes): An array of 32-bit unsigned integers representing the handles to cursors with the number of elements equal to the number of categories in the **CategorizationSet** field of CPMCreateQueryIn message plus one element, representing an uncategorized cursor.

2.2.3.6 CPMGetQueryStatusIn

The CPMGetQueryStatusIn message requests the status of a query. The format of the CPMGetQueryStatusIn message that follows the header is shown in the following diagram.

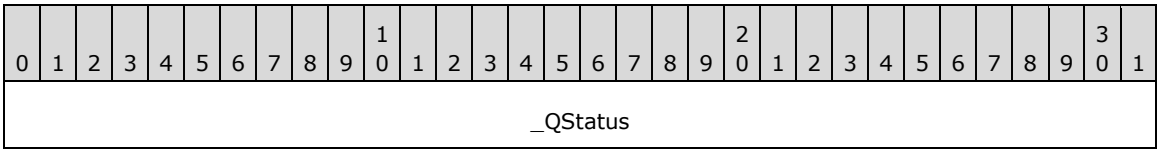


_hCursor (4 bytes): A 32-bit unsigned integer representing the handle from the CPMCreateQueryOut message identifying the query for which to retrieve status information.

2.2.3.7 CPMGetQueryStatusOut

The CPMGetQueryStatusOut message replies to a CPMGetQueryStatusIn message with the status of the query.

The format of the CPMGetQueryStatusOut message that follows the header is shown in the following diagram.



_QStatus (4 bytes): A 32-bit unsigned integer. A bitmask of values defined in the following tables that describe the query.

The following table lists STAT_* values obtained by performing a bitwise AND operation on **_Status** with 0x00000007. The result MUST be one of the following.

Constant	Meaning
STAT_BUSY 0x00000000	The asynchronous query is still running.
STAT_ERROR 0x00000001	The query is in an error state.
STAT_DONE 0x00000002	The query is complete and rows can be requested.

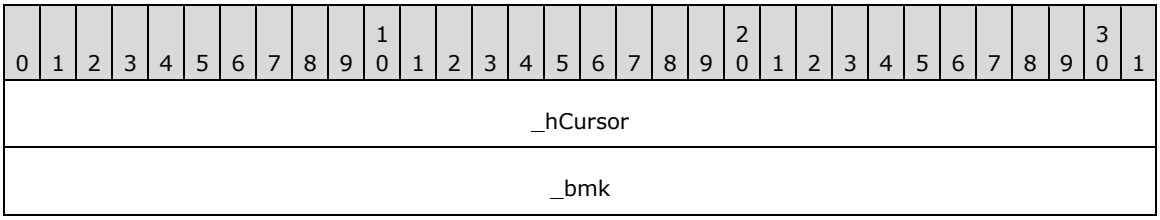
Constant	Meaning
STAT_REFRESH 0x00000003	The query is complete, but updates are resulting in additional query computation.

The following table lists additional STAT_* bits that can be set independently.

Constant	Meaning
STAT_NOISE_WORDS 0x00000010	Noise words were replaced by wildcard characters in the content query.
STAT_CONTENT_OUT_OF_DATE 0x00000020	The results of the query might be incorrect because the query involved modified but unindexed files.
STAT_CONTENT_QUERY_INCOMPLETE 0x00000080	The content query was too complex to complete or required enumeration instead of use of the content index.
STAT_TIME_LIMIT_EXCEEDED 0x00000100	The results of the query might be incorrect because the query execution reached the maximum allowable time.

2.2.3.8 CPMGetQueryStatusExIn

The CPMGetQueryStatusExIn message requests the status of a query and additional information, such as the number of documents that have been indexed or the number of documents remaining to be indexed. The format of the CPMGetQueryStatusExIn message that follows the header is shown in the following diagram.



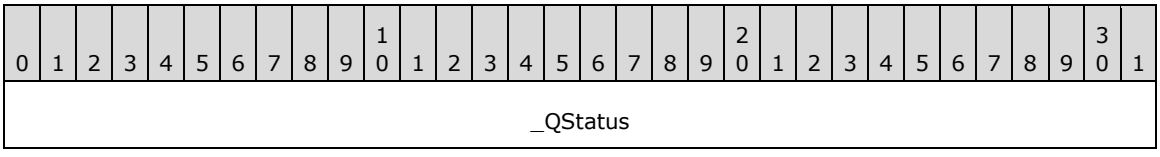
_hCursor (4 bytes): A 32-bit value representing the handle from the CPMCreateQueryOut message identifying the query for which to retrieve status information.

_bmk (4 bytes): A 32-bit value indicating the handle of a bookmark whose position is to be retrieved.

2.2.3.9 CPMGetQueryStatusExOut

The CPMGetQueryStatusExOut message replies to a CPMGetQueryStatusExIn message with both the status of the query and other status information, as outlined in the following diagram.

The format of the CPMGetQueryStatusExOut message that follows the header is shown in the following diagram.



_cFilteredDocuments
_cDocumentsToFilter
_dwRatioFinishedDenominator
_dwRatioFinishedNumerator
_iRowBmk
_cRowsTotal
_maxRank
_cResultsFound
_whereID

_QStatus (4 bytes): One of the STAT_* values specified in section 2.2.3.7.

_cFilteredDocuments (4 bytes): A 32-bit unsigned integer indicating the number of documents that have been indexed.

_cDocumentsToFilter (4 bytes): A 32-bit unsigned integer indicating the number of documents that remain to be indexed.

_dwRatioFinishedDenominator (4 bytes): A 32-bit unsigned integer indicating the denominator of the ratio of documents that the query has finished processing.

_dwRatioFinishedNumerator (4 bytes): A 32-bit unsigned integer indicating the numerator of the ratio of documents that the query has finished processing.

_iRowBmk (4 bytes): A 32-bit unsigned integer indicating the approximate position of the bookmark in the rowset in terms of rows.

_cRowsTotal (4 bytes): A 32-bit unsigned integer specifying the total number of rows in the rowset.

_maxRank (4 bytes): A 32-bit unsigned integer specifying the maximum rank found in the rowset.

_cResultsFound (4 bytes): A 32-bit unsigned integer specifying the number of unique results returned in the rowset.

_whereID (4 bytes): A 32-bit unsigned integer that defines a unique WHEREID for referring to the CRestrictionArray used to construct the rowset. This restriction can be reused as a restriction in future queries as long as there is still a cursor returned by CPMCreateQueryOut that has not been freed using CPMFreeCursorIn. This provides the server the option of sharing the evaluation of the restriction across queries.

2.2.3.10 CPMSetBindingsIn

The CPMSetBindingsIn message requests the binding of columns to a rowset. The server will reply to the CPMSetBindingsIn request message using the header section of the CPMSetBindingsIn message with the results of the request contained in the **_status** field. The format of the CPMSetBindingsIn message that follows the header is shown in the following diagram.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_hCursor (optional)																															
_cbRow (optional)																															
_cbBindingDesc (optional)																															
_dummy (optional)																															
cColumns (optional)																															
aColumns (variable)																															
...																															

_hCursor (4 bytes): A 32-bit value representing the handle from the CPMCreateQueryOut message that identifies the query for which to set bindings. This field MUST be present when the message is sent by the client and MUST be absent when the message is sent by the server.

_cbRow (4 bytes): A 32-bit unsigned integer indicating the size, in bytes, of a row. This field MUST be present when the message is sent by the client and MUST be absent when the message is sent by the server.

_cbBindingDesc (4 bytes): A 32-bit unsigned integer indicating the length, in bytes, of the fields following the **_dummy** field. This field MUST be present when the message is sent by the client and MUST be absent when the message is sent by the server.

_dummy (4 bytes): This field is unused and MUST be ignored. It can be set to any arbitrary value. This field MUST be present when the message is sent by the client and MUST be absent when the message is sent by the server.

cColumns (4 bytes): A 32-bit unsigned integer indicating the number of elements in the **aColumns** array. This field MUST be present when the message is sent by the client and MUST be absent when the message is sent by the server.

aColumns (variable): An array of CTableColumn structures describing the columns of a row in the rowset. This field MUST be present when the message is sent by the client and MUST be absent when the message is sent by the server. Structures in the array MUST be separated by 0 to 3 padding bytes such that each structure has a 4-byte alignment from the beginning of a message. Such padding bytes can be set to any arbitrary value when sent and MUST be ignored on receipt.

2.2.3.11 CPMGetRowsIn

The CPMGetRowsIn message requests rows from a query. The format of the CPMGetRowsIn message that follows the header is shown in the following diagram.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_hCursor																															
_cRowsToTransfer																															

_cbRowWidth
_cbSeek
_cbReserved
_cbReadBuffer
_ulClientBase
_fBwdFetch
eType
_chapt
SeekDescription (variable)
...

_hCursor (4 bytes): A 32-bit value representing the handle from the CPMCreateQueryOut message identifying the query for which to retrieve rows.

_cRowsToTransfer (4 bytes): A 32-bit unsigned integer indicating the maximum number of rows that the client will receive in response to this message.

_cbRowWidth (4 bytes): A 32-bit unsigned integer indicating the length of a row, in bytes.

_cbSeek (4 bytes): A 32-bit unsigned integer indicating the size of the message beginning with **eType**.

_cbReserved (4 bytes): A 32-bit unsigned integer indicating the size, in bytes, of a CPMGetRowsOut message (without the **Rows** and **SeekDescriptions** fields). This value in this field is added to the value of the **_cbSeek** field, and then is to be used to calculate the offset of **Rows** field in the CPMGetRowsOut message.

_cbReadBuffer (4 bytes): A 32-bit unsigned integer.

Note This field MUST be set to the maximum of the value of **_cbRowWidth** or 1000 times the value of **_cRowsToTransfer**, rounded up to the nearest 512 byte multiple. The value MUST NOT exceed 0x00004000.

_ulClientBase (4 bytes): A 32-bit unsigned integer indicating the base value to use for pointer calculations in the row buffer. If 64-bit offsets are being used, the **reserved2** field of the message header is used as the upper 32-bits and _ulClientBase as the lower 32-bits of a 64-bit value. See section 2.2.3.12.

_fBwdFetch (4 bytes): A 32-bit unsigned integer indicating the order in which to fetch the rows that MUST be set to one of the following values.

Value	Meaning
0x00000000	The rows are to be fetched in forward order.
0x00000001	The rows are to be fetched in reverse order.

eType (4 bytes): A 32-bit unsigned integer that MUST contain one of the following values indicating the type of operation to perform.

Value	Meaning
0x00000000	There is no **SeekDescription**; the **SeekDescription** field is omitted.
eRowSeekNext 0x00000001	**SeekDescription** contains a CRowSeekNext structure.
eRowSeekAt 0x00000002	**SeekDescription** contains a CRowSeekAt structure.
eRowSeekAtRatio 0x00000003	**SeekDescription** contains a CRowSeekAtRatio structure.
eRowSeekByBookmark 0x00000004	**SeekDescription** contains a CRowSeekByBookmark structure.

_chapt (4 bytes): A 32-bit value representing the handle of the rowset chapter.

SeekDescription (variable): This field MUST contain a structure of the type indicated by the **eType** value.

The *_fBwdFetch* argument affects the retrieval of rows from the results. Rows are taken from the beginning of the seek in the direction determined by the value of *_fBwdFetch*. In the following, a number of records have been retrieved. The figure shows how the buffer will be filled depending on the value of the *_fBwdFetch*. For more information about the structure of the buffer, see CPMGetRowsOut.

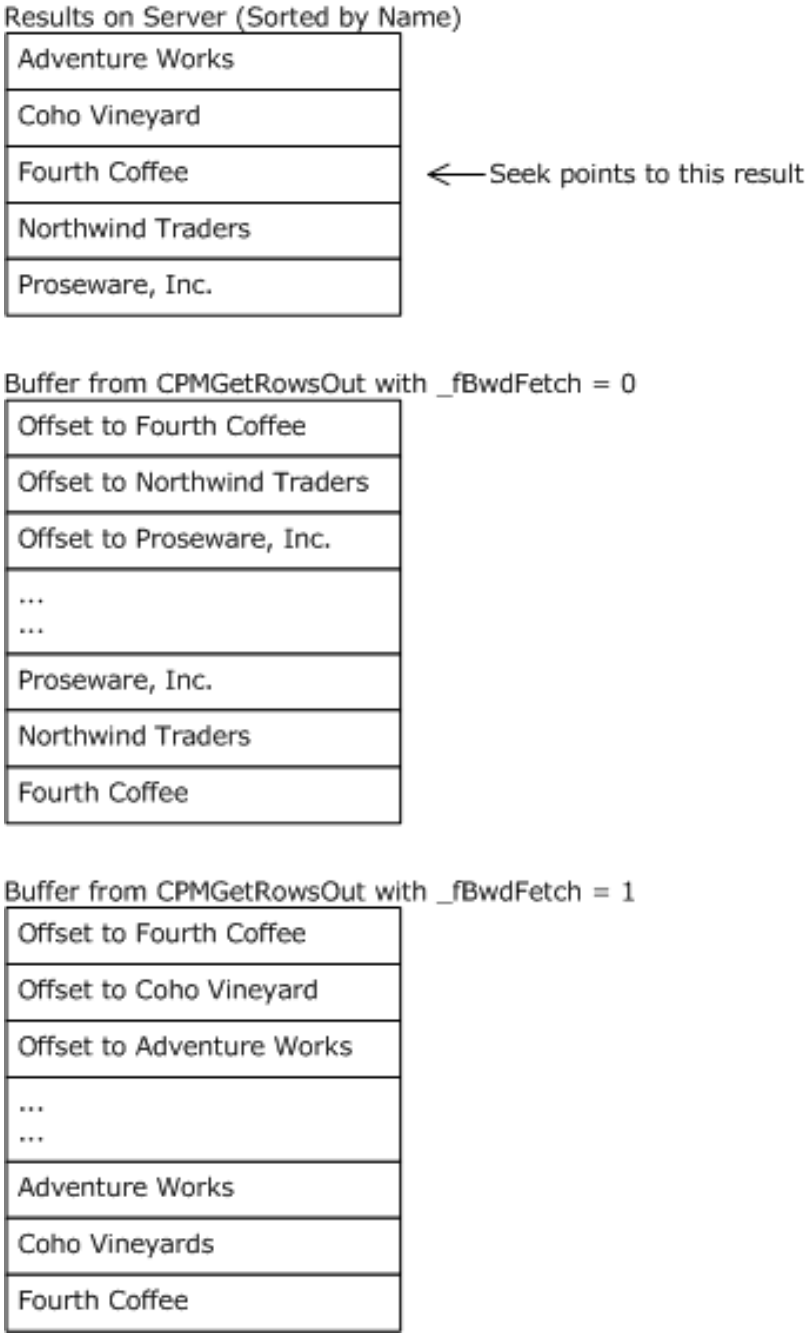


Figure 2: Effect of _fBwdFetch Parameter

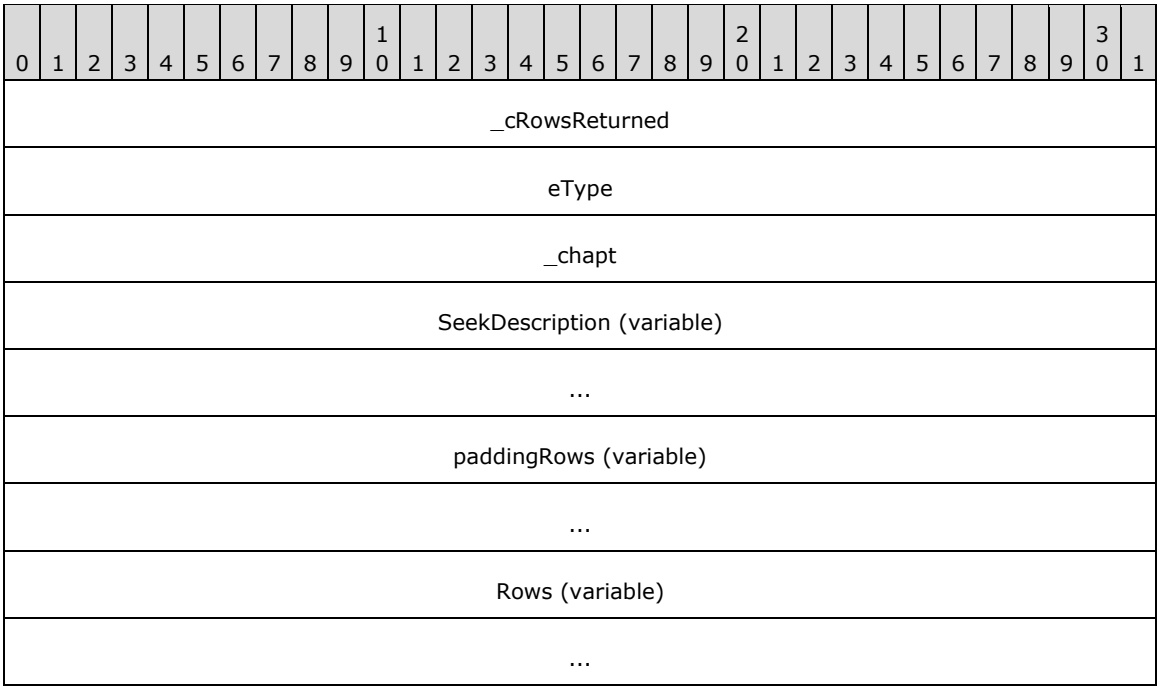
Notice that the *_fBwdFetch* argument changes the order in which records are presented for placement in the buffer but does not change the order (forward order) in which the buffer itself is filled.

2.2.3.12 CPMGetRowsOut

The CPMGetRowsOut message replies to a CPMGetRowsIn message with the rows of a query. Servers MUST format offsets to variable length data types in the row field as follows:

- The client indicated that it was a 32-bit system (**_iClientVersion** less than 0x00010000 in the **_iClientVersion** field of CPMConnectIn), and the server indicated that it was a 32-bit system (**_serverVersion** less than 0x00010000 in CPMConnectOut). Offsets are 32-bit integers.
- The client indicated that it was a 64-bit system (**_iClientVersion** greater than 0x00010000 in CPMConnectIn), and the server indicated that it was a 32-bit system (**_serverVersion** less than 0x00010000 in CPMConnectOut). Offsets are 32-bit integers.
- The client indicated that it was a 32-bit system (**_iClientVersion** less than 0x00010000 in the **_iClientVersion** field of CPMConnectIn), and the server indicated that it was a 64-bit system (**_serverVersion** greater than 0x00010000 in CPMConnectOut). Offsets are 32-bit integers.
- The client indicated that it was a 64-bit system (**_iClientVersion** greater than 0x00010000 in CPMConnectIn), and the server indicated that it was a 64-bit system (**_serverVersion** greater than 0x00010000 in CPMConnectOut). Offsets are 64-bit integers.

The format of the CPMGetRowsOut message that follows the header is shown in the following diagram.



_cRowsReturned (4 bytes): A 32-bit unsigned integer indicating the number of rows returned in **Rows**.

eType (4 bytes): A 32-bit unsigned integer. MUST contain one of the following values indicating the point at which to begin retrieving rows.

Value	Meaning
eRowsSeekNone 0x00000000	The **SeekDescription** is absent.
eRowSeekNext 0x00000001	**SeekDescription** contains a CRowSeekNext structure.
eRowSeekAt 0x00000002	**SeekDescription** contains a CRowSeekAt structure.

Value	Meaning
eRowSeekAtRatio 0x00000003	**SeekDescription** contains a CRowSeekAtRatio structure.
eRowSeekByBookmark 0x00000004	**SeekDescription** contains a CRowSeekByBookmark structure.

_chapt (4 bytes): A 32-bit value representing the handle of the rowset chapter.

SeekDescription (variable): This field MUST contain a structure of the type indicated by the **eType** field.

paddingRows (variable): This field MUST be of sufficient length (0 to _cbReserved-1 bytes) to pad the **Rows** field to _cbReserved offset from the beginning of a message, where _cbReserved is the value in the CPMGetRowsIn message. Padding bytes used in this field can be any arbitrary value. This field MUST be ignored by the receiver.

Rows (variable): Row data is formatted as prescribed by column information in the most recent CPMSetBindingsIn message. Rows MUST be stored in forward order (for example, row 1 before row 2).

Fixed-sized columns MUST be stored at the offsets specified by the most recent CPMSetBindingsIn message.

Variable-sized columns (for example, strings) MUST be stored as follows:

- The variable data itself (for example, the string) is stored near the end of the buffer in descending order (for example, the collection of all variable data for row 1 is at the end, row 2 next closest, and so on).
- The fixed-sized area (at the beginning of the row buffer) MUST contain a CTableVariant for each column, stored at the offset specified in the most recent CPMSetBindingsIn message. **vType** MUST contain the data type (for example, VT_LPWSTR). If, as determined by the rules at the beginning of this section, 32-bit offsets are being used, the **Offset** field in CTableVariant MUST contain a 32-bit value that is the offset of the variable data from the beginning of the CPMGetRowsOut message, plus the value of **_ulClientBase** specified in the most recent CPMGetRowsIn message. If 64-bit offsets are being used, the **Offset** field in CTableVariant MUST contain a 64-bit value that is the offset from the beginning of the CPMGetRowsOut message, added to a 64-bit value composed by using **_ulClientBase** as the low 32-bits and **_ulReserved2** as the high 32-bits.

The buffer is filled in from both ends. CTableVariant structures, one for each row, are stored at the beginning of the buffer. Each of these structures points to the row data which is stored starting at the end of the buffer.

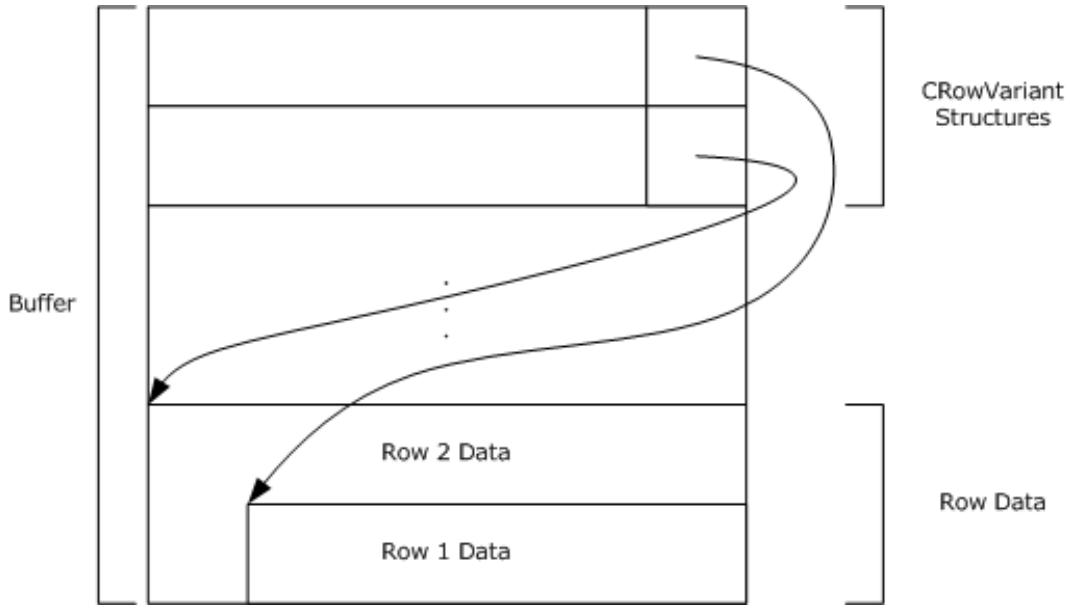


Figure 3: Structure of the row buffer

2.2.3.13 CPMRatioFinishedIn

The CPMRatioFinishedIn message requests the completion percentage of a query. The format of the CPMRatioFinishedIn message that follows the header is shown in the following diagram.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_hCursor																															
_fQuick																															

_hCursor (4 bytes): The handle from the CPMCreateQueryOut message identifying the query for which to request completion information.

_fQuick (4 bytes): This is unused and MUST be ignored by the server.

Note This field MUST be set to 0x00000001.

2.2.3.14 CPMRatioFinishedOut

The CPMRatioFinishedOut message replies to a CPMRatioFinishedIn message with the completion ratio of a query. The format of the CPMRatioFinishedOut message that follows the header is shown in the following diagram.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
_ulNumerator																															
_ulDenominator																															

_cRows
_fNewRows

_ulNumerator (4 bytes): A 32-bit unsigned integer indicating the numerator of the completion ratio in terms of rows.

_ulDenominator (4 bytes): A 32-bit unsigned integer indicating the denominator of the completion ratio in terms of row.<13>

_cRows (4 bytes): A 32-bit unsigned integer indicating the total number of rows for the query.

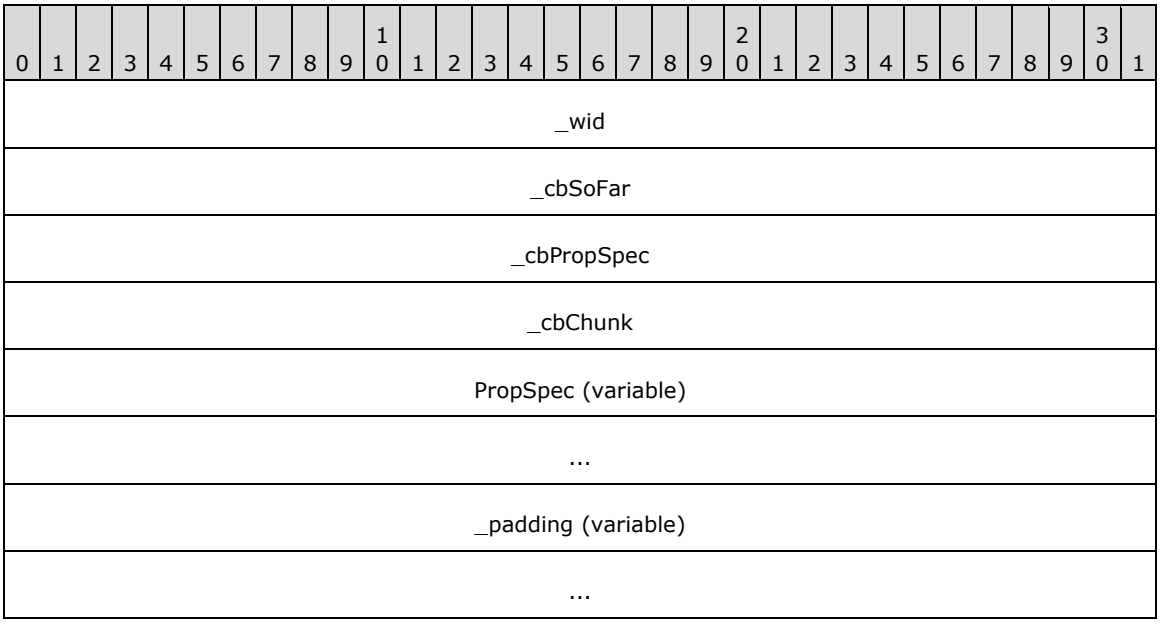
_fNewRows (4 bytes): A Boolean value indicating if there are new rows available. This field MUST NOT be set to any values other than the following.

Value	Meaning
0x00000000	There are no new rows in the rowset.
0x00000001	There are new rows available in the rowset.

2.2.3.15 CPMFetchValueIn

The CPMFetchValueIn message requests a property value that was too large to return in a rowset. As specified in section 3.2.4.2.5, this message is sent repeatedly to retrieve all bytes of the property, updating **_cbSoFar** for each, until the **_fMoreExists** field of the CPMFetchValueOut message is set to FALSE.

The format of the CPMFetchValueIn message that follows the header is shown in the following diagram.



_wid (4 bytes): A 32-bit unsigned integer representing the document ID identifying the document for which a property is to be fetched.

_cbSoFar (4 bytes): A 32-bit unsigned integer containing the number of bytes previously transferred for this property.

Note This field MUST be set to 0x00000000 in the first message.

_cbPropSpec (4 bytes): A 32-bit unsigned integer containing the size, in bytes, of the **PropSpec** field.

_cbChunk (4 bytes): A 32-bit unsigned integer containing the maximum number of bytes that the sender can accept in a CPMFetchValueOut message.<14>

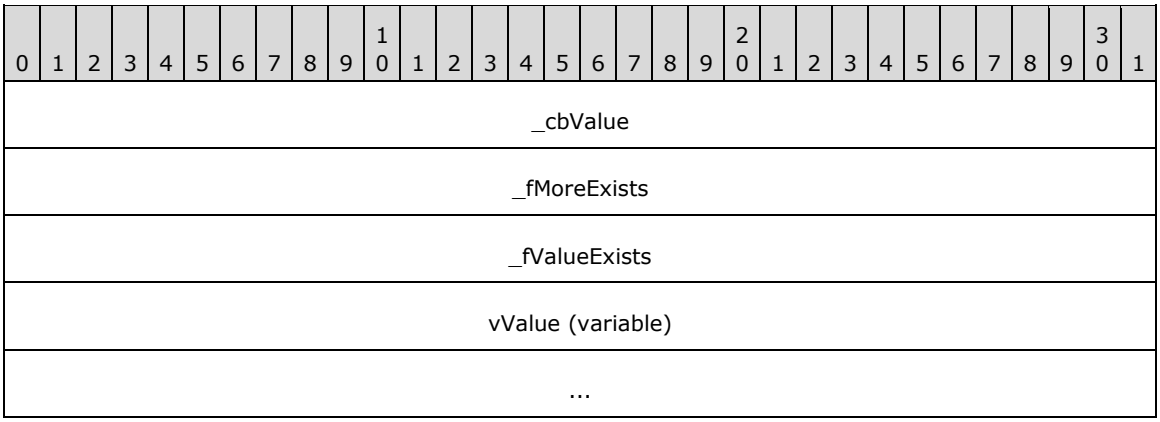
PropSpec (variable): A CFullPropSpec structure specifying the property to retrieve.

_padding (variable): This field MUST be of the length necessary (0 to 3 bytes) to pad the message out to a multiple of 4 bytes in length. The value of the padding bytes can be any arbitrary value. This field MUST be ignored by the receiver.

2.2.3.16 CPMFetchValueOut

The CPMFetchValueOut message replies to a CPMFetchValueIn message with a property value from a previous query. As specified in section 3.2.4.2.5, this message is sent after each CPMFetchValueIn message until all bytes of the property are transferred.

The format of the CPMFetchValueOut message that follows the header is shown in the following diagram.



_cbValue (4 bytes): A 32-bit unsigned integer containing the total size, in bytes, of **vValue**.

_fMoreExists (4 bytes): A Boolean value indicating whether additional CPMFetchValueOut messages are available.

Value	Meaning
0x00000000	There are no additional data available.
0x00000001	There are additional data available.

_fValueExists (4 bytes): A Boolean value indicating whether there is a value for the property.

Value	Meaning
0x00000000	A value for the property does not exist.
0x00000001	A value for the property exists.

vValue (variable): A portion of a byte array containing a SERIALIZEDPROPERTYVALUE, where the offset of the beginning of the portion is the value of **_cbSoFar** in CPMFetchValueIn. The length of the portion, indicated by the **_cbValue** field, MUST be less than or equal to the value of **_cbChunk** in CPMFetchValueIn.

2.2.3.17 CPMGetNotify

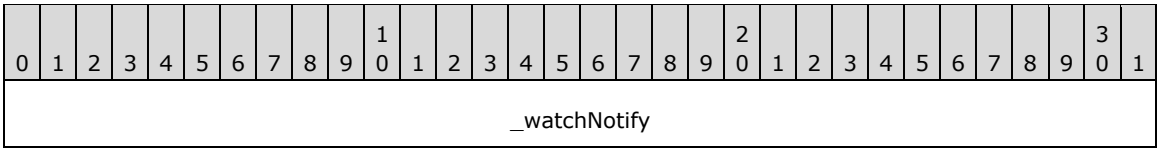
The CPMGetNotify message requests that the client wants to be notified of rowset changes.

The message MUST NOT include a body; only the message header, as specified in section 2.2.2, is sent.

2.2.3.18 CPMSendNotifyOut

The CPMSendNotifyOut message notifies the client of a change to the results of a query.

This message is only sent when a change occurs. The format of the CPMSendNotifyOut message that follows the header is shown in the following diagram.

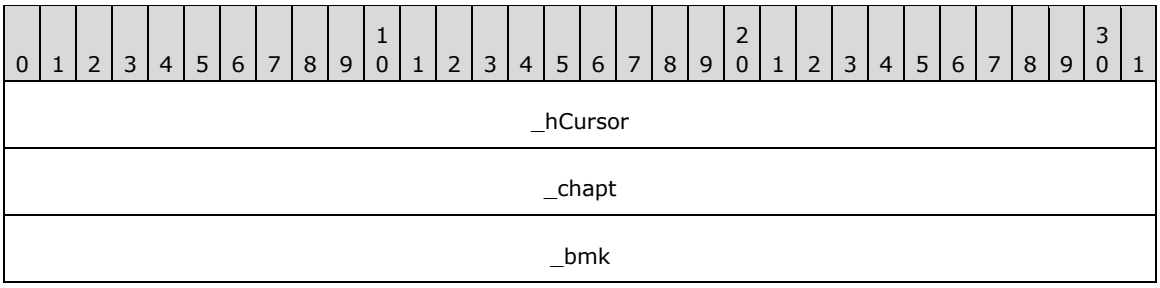


_watchNotify (4 bytes): A 32-bit unsigned integer representing the change to the query. It MUST be one of the following values.<15>

Value	Meaning
DBWATCHNOTIFY_ROWSCHANGED 0x00000001	The number of rows in the query rowset has changed.
DBWATCHNOTIFY_QUERYDONE 0x00000002	The query has completed.
DBWATCHNOTIFY_QUERYREEXECUTED 0x00000003	The query has been executed again.

2.2.3.19 CPMGetApproximatePositionIn

The CPMGetApproximatePositionIn message requests the approximate position of a bookmark in a chapter. The format of the CPMGetApproximatePositionIn message that follows the header is shown in the following diagram.<16>



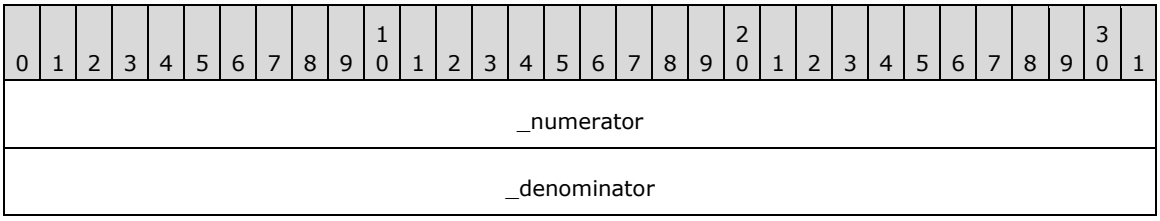
_hCursor (4 bytes): A 32-bit value representing the query cursor obtained from CPMCreateQueryOut for the rowset containing the bookmark.

_chapt (4 bytes): A 32-bit value representing the handle to the chapter containing the bookmark.

_bmk (4 bytes): A 32-bit value representing the handle to the bookmark for which to retrieve the approximate position.

2.2.3.20 CPMGetApproximatePositionOut

The CPMGetApproximatePositionOut message replies to a CPMGetApproximatePositionIn message describing the approximate position of the bookmark in the chapter. The format of the CPMGetApproximatePositionOut message that follows the header is shown in the following diagram.



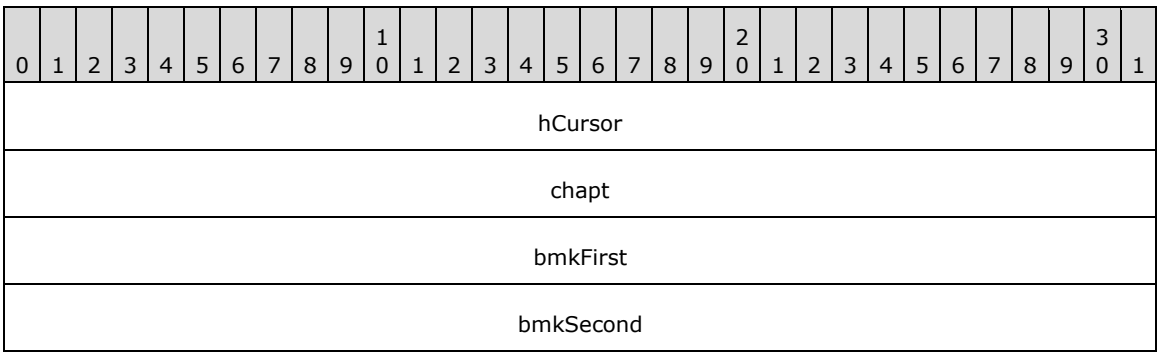
_numerator (4 bytes): A 32-bit unsigned integer containing the row number of the bookmark in the rowset. If there are no rows, this field MUST be set to 0x00000000.

_denominator (4 bytes): A 32-bit unsigned integer containing the number of rows in the rowset.

2.2.3.21 CPMCompareBmkIn

The CPMCompareBmkIn message requests a comparison of two bookmarks in a chapter.<17>

The format of the CPMCompareBmkIn message that follows the header is shown in the following diagram.



hCursor (4 bytes): A 32-bit unsigned integer representing the handle from the CPMCreateQueryOut message for the rowset containing the bookmarks.

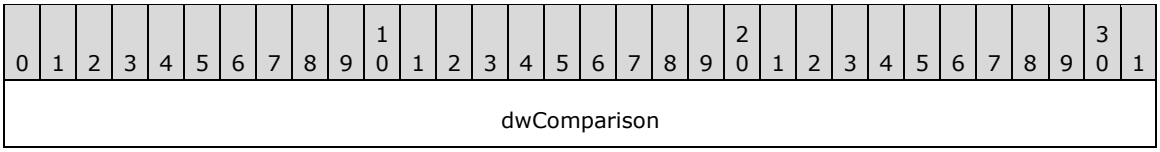
chapt (4 bytes): A 32-bit unsigned integer representing the handle of the chapter containing the bookmarks to compare.

bmkFirst (4 bytes): A 32-bit unsigned integer representing the handle to the first bookmark to compare.

bmkSecond (4 bytes): A 32-bit unsigned integer representing the handle to the second bookmark to compare.

2.2.3.22 CPMCompareBmkOut

The CPMCompareBmkOut message replies to a CPMCompareBmkIn message with the comparison of the two bookmarks in the chapter. The format of the CPMCompareBmkOut message that follows the header is shown in the following diagram.



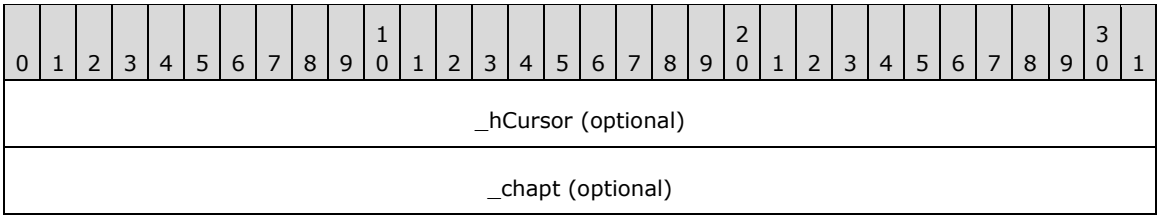
dwComparison (4 bytes): A 32-bit unsigned integer. MUST be one of the following values, indicating the relative positions of the two bookmarks in the chapter.

Value	Meaning
DBCOMPARE_LT 0x00000000	The first bookmark is positioned before the second.
DBCOMPARE_EQ 0x00000001	The first bookmark has the same position as the second.
DBCOMPARE_GT 0x00000002	The first bookmark is positioned after the second.
DBCOMPARE_NE 0x00000003	The first bookmark does not have the same position as the second.
DBCOMPARE_NOTCOMPARABLE 0x00000004	The first bookmark is not comparable to the second.

2.2.3.23 CPMRestartPositionIn

The CPMRestartPositionIn message moves the fetch position for a cursor to the beginning of the chapter. As specified in section 3.1.5.2.12, the server will reply using the same message, with the results of the request contained in the **_status** field of the Windows Search Protocol header. <18>

The format of the CPMRestartPositionIn message that follows the header is shown in the following diagram.

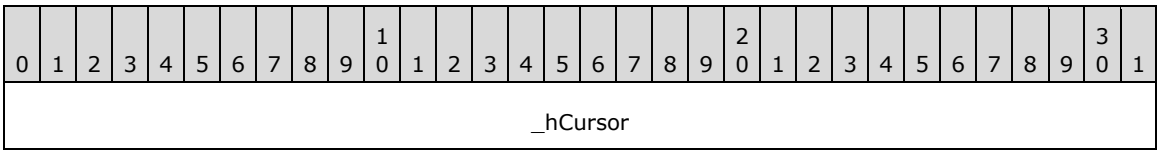


_hCursor (4 bytes): A 32-bit value that represents the handle obtained from a CPMCreateQueryOut message and that identifies the query for which to restart the position. This field MUST be present when the message is sent by the client and MUST be absent when the message is sent by the server.

_chapt (4 bytes): A 32-bit value representing the handle of a chapter from which to retrieve rows. This field MUST be present when the message is sent by the client and MUST be absent when the message is sent by the server.

2.2.3.24 CPMFreeCursorIn

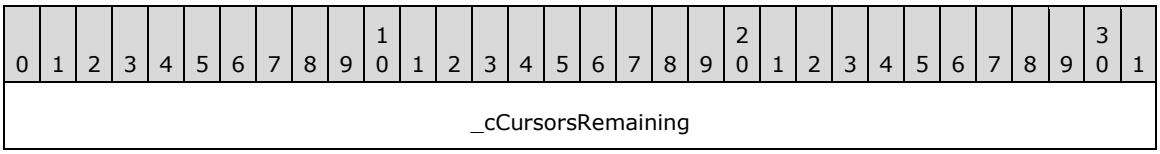
The CPMFreeCursorIn message requests the release of a cursor. The format of the CPMFreeCursorIn message that follows the header is shown in the following diagram.



_hCursor (4 bytes): A 32-bit value representing the handle of the cursor from the CPMCreateQueryOut message to release.

2.2.3.25 CPMFreeCursorOut

The CPMFreeCursorOut message replies to a CPMFreeCursorIn message with the results of freeing a cursor. The format of the CPMFreeCursorOut message that follows the header is shown in the following diagram.



_cCursorsRemaining (4 bytes): A 32-bit unsigned integer indicating the number of cursors still in use for the query.

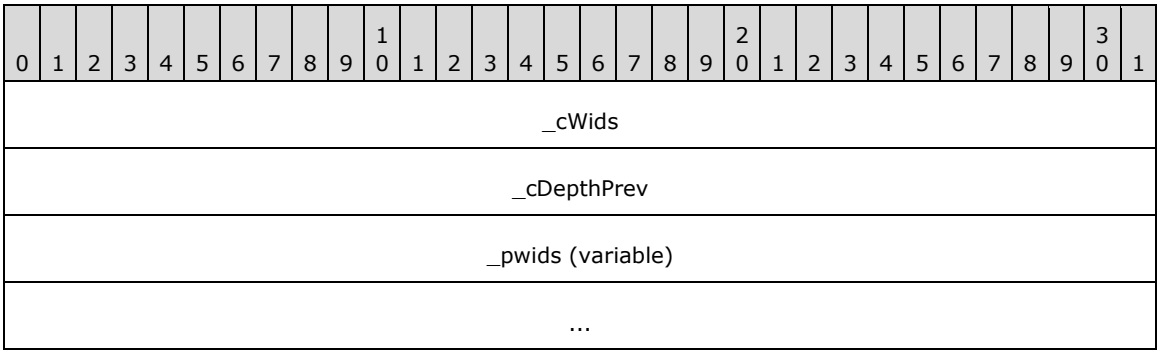
2.2.3.26 CPMDisconnect

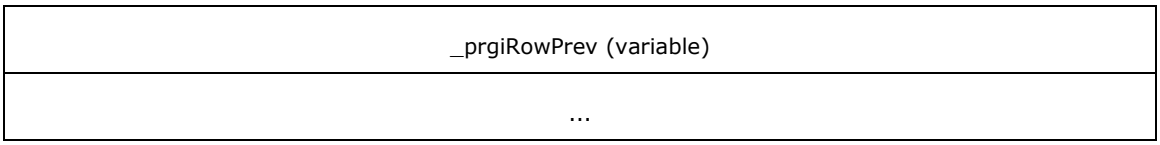
The CPMDisconnect message ends the connection with the server.

The message MUST NOT include a body; only the message header, as specified in section 2.2.2, is sent.

2.2.3.27 CPMFindIndicesIn

The CPMFindIndicesIn <19> message requests the rowset position of the next occurrence of a document identifier.





_cWids (4 bytes): A 32-bit unsigned integer representing the number of document identifiers to search for. This value MUST be greater than zero.

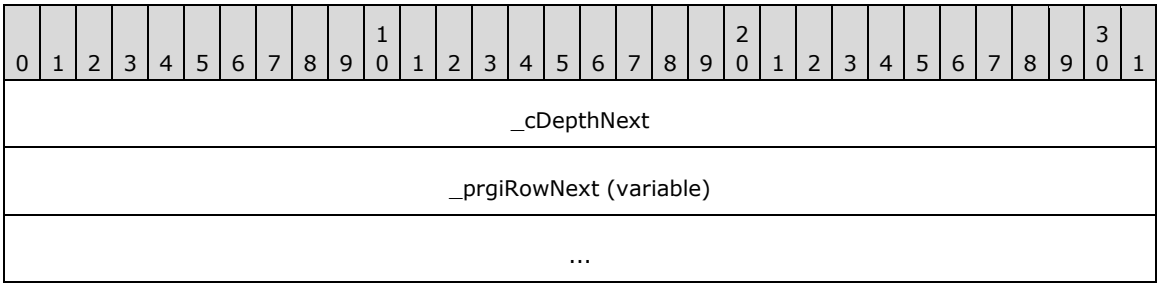
_cDepthPrev (4 bytes): A 32-bit unsigned integer specifying the number of hierarchical grouping levels leading to the previous occurrence of any of the specified document identifiers. When this value is zero it instructs the server to begin searching at the beginning of the cursor.

_pwids (variable): Array of unsigned 32-bit integers containing the document identifiers to search for. The size of the array MUST be equal to **_cWids**.

_prgiRowPrev (variable): Array of unsigned 32-bit integers representing the hierarchical group indices leading to the previous occurrence of any of the specified document identifiers. The size of the array MUST be equal to **_cDepthPrev**. If **_cDepthPrev** is equal to zero this field MUST be omitted.

2.2.3.28 CPMFindIndicesOut

The CPMFindIndicesOut message replies to a CPMFindIndicesIn message with the hierarchical group indices leading to the next occurrence of any of the document identifiers specified in the CPMFindIndicesIn message.



_cDepthNext (4 bytes): A 32-bit unsigned integer specifying the number of hierarchical grouping levels leading to the next occurrence of any of the document identifiers specified in the corresponding CPMFindIndicesIn message. This value MUST be set to zero if no occurrence of document identifier has been found following the position of hierarchical group indices specified by preceding CPMFindIndicesIn message.

_prgiRowNext (variable): Array of unsigned 32-bit integers representing the hierarchical grouping indices leading to the next occurrence of any of the document identifiers specified in the corresponding CPMFindIndicesIn message. The size of the array MUST be equal to **_cDepthNext**. This field MUST be omitted if **_cDepthNext** is equal to zero.

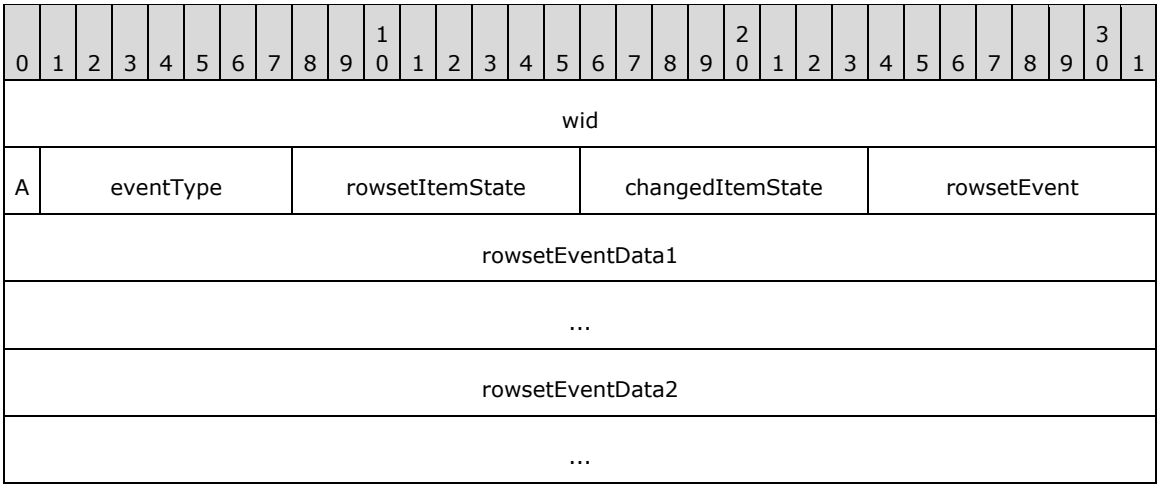
2.2.3.29 CPMGetRowsetNotifyIn

The CPMGetRowsetNotifyIn <20> message requests the next rowset event from the server if available.

The message MUST NOT include a body; only the message header, as specified in section 2.2.2.

2.2.3.30 CPMGetRowsetNotifyOut

The CPMGetRowsetNotifyOut message replies to CPMGetRowsetNotifyIn message with oldest available rowset event. The format of the CPMGetRowsetNotifyOut message that follows the header is shown in the following diagram.



wid (4 bytes): A 32-bit unsigned integer containing the document identifier that the event is for. This value MUST be zero if **eventType** is PROPAGATE_NONE or PROPAGATE_ROWSET.

A - moreEvents (1 bit): A single bit that is set to 1 only if there are additional rowset events remaining on the server.

eventType (7 bits): A 7 bit unsigned integer that MUST be one of the following values, indicating the type of event this message represents.

Value	Meaning
PROPAGATE_NONE 0	This response indicates that there were no available rowset events waiting on the server.
PROPAGATE_ADD 1	This response indicates that an item was added to the index that could be relevant to the query originating the rowset.
PROPAGATE_DELETE 2	This response indicates that an item was deleted from the index that could be relevant to the query originating the rowset.
PROPAGATE_MODIFY 3	This response indicates that an item was re-indexed that could be relevant to the query originating the rowset.
PROPAGATE_ROWSET 4	This response is a rowset specific notification whose meaning is interpreted by the rowsetEvent field of this message.

rowsetItemState (1 byte): An 8 bit unsigned integer that MUST be one of the following values if **eventType** is PROPAGATE_ADD, PROPAGATE_DELETE, or PROPAGATE_MODIFY. This number indicates the state of the document identifier specified by wid within the originating rowset. For other **eventType** values this value MUST be set to zero.

Value	Meaning
ROWSETEVENT_ITEMSTATE_NOTINROWSET 0	The document identifier specified by **wid** MUST not have been contained within the originating rowset.

Value	Meaning
ROWSETEVENT_ITEMSTATE_INROWSET 1	The document identifier speicified by wid MUST be contained within the originating rowset.
ROWSETEVENT_ITEMSTATE_UNKNOWN 2	The document identifier speicifed by wid's containment within the originating rowset has not been specified.

changedItemState (1 byte): An 8 bit unsigned integer that MUST be one of the following values if **eventType** is PROPAGATE_MODIFY. This number indicates the state of the document identifier specified by wid within the originating rowset if the same query were to be run again following the change. For other **eventType** values this value MUST be set to zero.

Value	Meaning
ROWSETEVENT_ITEMSTATE_NOTINROWSET 0	The document identifier specified by wid would NOT be contained within a subsequent query.
ROWSETEVENT_ITEMSTATE_INROWSET 1	The document identifier speicified by **wid** would be contained within a subsequent query.
ROWSETEVENT_ITEMSTATE_UNKNOWN 2	Whether or not the document identifier speicifed by wid would be contained within a subsequent query has not been specified.

rowsetEvent (1 byte): An 8 bit unsigned integer that MUST be one of the following values if **eventType** is PROPAGATE_ROWSET. This number indicates the type of rowset event that this message represents. For other **eventType** values this value MUST be set to zero.

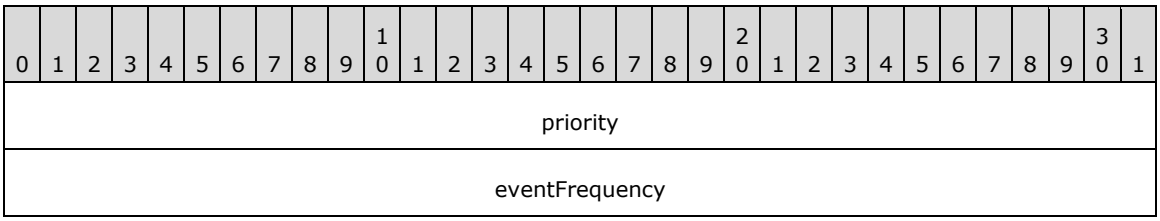
Value	Meaning
ROWSETEVENT_TYPE_DATAEXPIRED 0	The data backing the rowset is no longer valid. RowsetEventData1 and RowsetEventData2 MUST be set to zero.
ROWSETEVENT_TYPE_FOREGROUNDLOST 1	The rowset no longer has foreground priority and has been reverted to high priority. Items that apply to this query will be indexed at a decreased rate. See section 2.2.3.34 for meaning of foreground and high priority. RowsetEventData1 and RowsetEventData2 MUST be set to zero.
ROWSETEVENT_TYPE_SCOPESTATISTICS 2	The number of indexed items, number of items that need to be indexed, or number of items that need to be re-indexed has changed. RowsetEventData1's high 32 bits contain a 32-bit unsigned integer indicating the number of items that need to be indexed that could be relevant to the originating rowset. **RowsetEventData1**'s low 32 bits contain a 32-bit unsigned integer indicating the number of items that need to be re-indexed that could be relevant to the originating rowset. RowsetEventData2's high 32 bits MUST be set to zero. **RowsetEventData2**'s low 32 bits contain a 32-bit unsigned integer indicating the number of indexed items that could be relevant to the originating rowset.

rowsetEventData1 (8 bytes): A 64 bit unsigned number whose meaning is dependent on **rowsetEvent**. Undefined unless **eventType** is PROPAGATE_ROWSET.

rowsetEventData2 (8 bytes): A 64 bit unsigned number whose meaning is dependent on **rowsetEvent**. Undefined unless **eventType** is PROPAGATE_ROWSET.

2.2.3.31 CPMSetScopePrioritizationIn

The CPMSetScopePrioritizationIn<21> message requests that the server prioritize indexing of items that could be relevant to the originating query at a rate specified in the message. The format of the CPMSetScopePrioritizationIn message that follows the header is shown in the following diagram.



priority (4 bytes): A 32-bit unsigned integer containing the type of prioritization requested for documents that could be relevant to the originating query.

Value	Meaning
PRIORITY_LEVEL_FOREGROUND 0	Process items that could be relevant to the originating query before others as quickly as possible.
PRIORITY_LEVEL_HIGH 1	Process items that could be relevant to the originating query before others at the normal rate.
PRIORITY_LEVEL_LOW 2	Process items that could be relevant to the originating query before others, but after any other prioritization requests at the normal rate.
PRIORITY_LEVEL_DEFAULT 3	Process items at the normal rate.

eventFrequency (4 bytes): A 32-bit unsigned integer containing the minimum suggested interval in milliseconds between subsequent issued rowset events of the type ROWSETEVENT_TYPE_SCOPESTATISTICS. When eventFrequency is set to zero the server MUST NOT issue the ROWSETEVENT_TYPE_SCOPESTATISTICS events. This field MUST be set to zero when setting priority to PRIORITY_LEVEL_DEFAULT.

2.2.3.32 CPMSetScopePrioritizationOut

The CPMSetScopePrioritizationOut message replies to the CPMSetScopePrioritizationIn message.

The message MUST NOT include a body; only the message header, as specified in section 2.2.2, is sent.

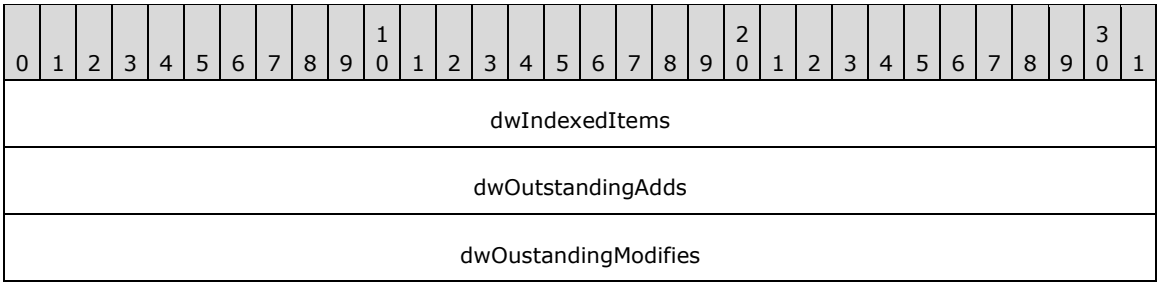
2.2.3.33 CPMGetScopeStatisticsIn

The CPMGetScopeStatisticsIn <22> message requests statistics regarding the number of indexed items, the number of items needing to be indexed and the number of items needing to be re-indexed that are relevant to the originating query.

The message MUST NOT include a body; only the message header, as specified in section 2.2.2, is sent.

2.2.3.34 CPMGetScopeStatisticsOut

The CPMGetScopeStatisticsOut message replies to CPMGetScopeStatisticsIn message with the requested statistics for the originating rowset. The format of the CPMGetScopeStatisticsOut message that follows the header is shown in the following diagram.



dwIndexedItems (4 bytes): A 32-bit unsigned integer containing the number of items that are currently indexed that are relevant to the originating query.

dwOutstandingAdds (4 bytes): A 32-bit unsigned integer containing the number of items that have yet to be indexed that could be relevant to the originating query.

dwOustandingModifies (4 bytes): A 32-bit unsigned integer containing the number of items that need to be re-indexed that are relevant to the originating query.

2.2.4 Errors

Windows Search Protocol (WSP) messages indicate success two ways:

- A zero value (0x00000000).
- An HRESULT success value, such as DB_S_ENDOFROWSET, in which the thirty-first bit is not set.

Otherwise, WSP messages return a 32-bit error code that can either be an HRESULT or an NTSTATUS value (see section 1.8). If a buffer is too small to fit a result, a status code of STATUS_INSUFFICIENT_RESOURCES (0xC0000009A) MUST be returned, and the failing operation can be retried with a larger buffer.

All other error values are treated the same; the error is considered fatal and reported to the higher-level caller. Future messages MAY be sent over the same pipe as if no error had occurred.<23>

The following are common error codes:

- E_OUTOFMEMORY (0x8007000e)
- STATUS_INVALID_PARAMETER (0xc000000d)
- STATUS_NO_MEMORY (0xc0000017)
- STATUS_INSUFFICIENT_RESOURCES (0xc000009a)
- CI_E_NOT_FOUND (0x80041815)
- STATUS_INVALID_PARAMETER_MIX (0xc0000030)
- ERROR_INVALID_PARAMETER (0x57)
- CI_E_TIMEOUT (0x8004181F)
- E_ACCESSDENIED (0x80070005)

- CI_E_BUFFERTOOSMALL (0x8004180c)

Note The HRESULT and NTSTATUS numbering spaces do not currently overlap—except with values of identical meaning. However, even if there are conflicts in the future, they would not cause protocol issues as long as the value for STATUS_INSUFFICIENT_RESOURCES remains unique, because all other error values are treated the same.

2.2.5 Standard Properties

Properties in the GSS are represented by the combination of a property set GUID and either a string property name or an integer property ID. For details, see CFullPropSpec.

There are three classes of properties: database properties, query properties, and open properties. Database properties help control the indexing behavior and are as specified in section 2.2.1.31.1. Query properties can be used in a restriction and in some cases returned with every result. They are special because they are built into the GSS. Open properties are defined by individual applications. There is a typical set of common properties in use, but there is no requirement to use them.

Properties are characterized by the parameters that follow.

GUID and PropId: Together, these parameters establish the unique identifier for documents.

isColumn: A boolean value set to TRUE if, and only if, the property can be returned as a requested property as specified in the *ProjectionColumnsOffsets* argument to a **RunNewQuery** Generic Search Service (GSS) abstract interface call.

inInvertedIndex: A boolean value set to TRUE if, and only if, the property can be an argument to CContentRestriction within the *RestrictionSet* argument to a **RunNewQuery** GSS abstract interface call.

columnIndexType: This parameter defines whether sorting, grouping, and filtering are allowed for this property, as defined in the *SortOrders*, *Groupings*, and *Restrictions* parameters of the **RunNewQuery** GSS abstract interface call. The *columnIndexType* parameter is a string set to one of the following:

- No value: indicates that sorting, grouping and filtering over this property are possible.
- NotIndexed: sorting, grouping, and filtering are not possible for this property.

Type: This parameter indicates the data type used by this property.

- Int32 – VT_I8
- String - LPWSTR
- Boolean – VT_BOOL
- UInt16 – VT_UI2
- UInt32 – VT_UI4
- UInt64 – VT_UI8
- Double – VT_R8
- DateTime – VT_FILETIME.
- Buffer – VT_BLOB_OBJECT
- Byte – VT_UI1

MaxSize: An unsigned integer indicating the maximum size, in bytes, that this property can have stored or retrieved by the Generic Search Service (GSS).

VectorProperty: A boolean value that is TRUE if, and only if, this is a multiple-value property. For example, System.Author can have multiple values, one for each author of the document.

Description: Describes the most common use for a property. The names of some properties are self-explanatory; when this is the case, the descriptions are omitted.

useForTypeAhead: A Boolean value that is TRUE if and only if this is a property whose values are used as a source of completion suggestions.

2.2.5.1 Query Properties

Query Property Set

```
#define  QueryGuid     \
{0x49691c90,0x7e17,0x101a,0xa9,0x1c,0x08,0x00,0x2b,0x2e,0xcd,0xa9}
```

Name/PropId	Datatype	Description
RankVector 0x00000002	VT_UI4|VT_VECTOR	The 0-1000 rank computed for each element when performing vector ranking.
System.Search.Rank 0x00000003	VT_I4	The rank 0-1000 computed for this item. How rank is computed is defined by the indexing process. Typically, content and proximity restrictions influence the rank, while other comparison operators do not.
System.Search.HitCount 0x00000004	VT_I4	The number of words found from the query.
System.Search.EntryID 0x00000005	VT_I4	A unique identifier for each result found.
All 0x00000006	VT_LPWSTR	Allows a content restriction over all textual properties. Cannot be retrieved.
System.ItemURL 0x00000009	VT_LPWSTR	Full virtual path to file, including file name.

Storage Property Set

```
#define  StorageGuid     \
{0xb725f130,0x47ef,0x101a,0xa5,0xf1,0x02,0x60,0x8c,0x9e,0xeb,0xac}
```

The friendly name is Contents, the PropId is 0x00000013, the data type is VT_LPWSTR, and it is the main contents of a file. Usually this property cannot be retrieved.

2.2.5.2 Common Open Properties

A GSS can allow querying and retrieval over any property. The tables below outline some properties typically used. See [MSDN-PROPLIST] for more information.

Storage Property Set

```
#define  StorageGuid     \
{0xb725f130,0x47ef,0x101a,0xa5,0xf1,0x02,0x60,0x8c,0x9e,0xeb,0xac}
```

Name/PropId	Type	Description
System.ItemFolderNameDisplay 0x00000002	VT_LPWSTR	The physical path to the file, not including the file name.
ClassId 0x00000003	VT_CLSID	The class ID of the object, for example, WordPerfect or Microsoft® Office Word.
FileIndex 0x00000008	VT_UI8	The unique ID of the file.
USN 0x00000009	VT_I8	The update sequence number. NTFS file system drives only.
System.ItemNameDisplay 0x0000000A	VT_LPWSTR	The name of the file.
Path 0x0000000B	VT_LPWSTR	The full physical path to the file, including the file name.
System.Size 0x0000000C	VT_I8	The size, in bytes, of the file.
System.FileAttributes 0x0000000D	VT_UI4	The file attributes. Documented in Win32 SDK.
System.DateModified 0x0000000E	VT_FILETIME	The last time the file was written.
System.DateCreated 0x0000000F	VT_FILETIME	The time the file was created.
System.DateAccessed 0x00000010	VT_FILETIME	The last time the file was accessed.
AllocSize 0x00000012	VT_I8	The size of the disk allocation for the file.
ShortFilename 0x00000014	VT_LPWSTR	The short (8.3) file name.

The following table lists the attribute flag values for the Attrib property.

Attribute/Value	Description
FILE_ATTRIBUTE_READONLY 0x00000001	The file or directory is read-only. Applications can read the file but cannot write to it or delete it. For a directory, applications cannot delete it.
FILE_ATTRIBUTE_HIDDEN 0x00000002	The file or directory is hidden. It is not included in an ordinary directory listing.
FILE_ATTRIBUTE_SYSTEM 0x00000004	The file or directory is part of the operating system, or is used exclusively by the operating system.

Attribute/Value	Description
FILE_ATTRIBUTE_DIRECTORY 0x00000010	The handle identifies a directory.
FILE_ATTRIBUTE_ARCHIVE 0x00000020	The file or directory is an archive file. Applications use this attribute to mark files for backup or removal.
FILE_ATTRIBUTE_NORMAL 0x00000080	The file or directory does not have another attribute set. This attribute is valid only if used alone.
FILE_ATTRIBUTE_TEMPORARY 0x00000100	The file is being used for temporary storage. File systems avoid writing data back to mass storage if sufficient cache memory is available, because often the application deletes the temporary file shortly after the handle is closed. In that case, the system can entirely avoid writing the data. Otherwise, the data is written after the handle is closed.
FILE_ATTRIBUTE_SPARSE_FILE 0x00000200	The file is a sparse file.
FILE_ATTRIBUTE_REPARSE_POINT 0x00000400	The file or directory has an associated reparse point.
FILE_ATTRIBUTE_COMPRESSED 0x00000800	The file or directory is compressed. For a file, this means that all the data in the file is compressed. For a directory, this means that compression is the default for newly created files and subdirectories.
FILE_ATTRIBUTE_OFFLINE 0x00001000	The data of the file is not immediately available. This attribute indicates that the file data has been physically moved to offline storage. This attribute is used by Remote Storage, the hierarchical storage management software. Applications SHOULD NOT arbitrarily change this attribute.
FILE_ATTRIBUTE_ENCRYPTED 0x00004000	The file or directory is encrypted. For a file, this means that all data in the file is encrypted. For a directory, this means that encryption is the default for newly created files and subdirectories.
FILE_ATTRIBUTE_VIRTUAL 0x00010000	A file is a virtual file.

Property Sets for Documents

```
#define  DocPropSetGuid      \
{0xf29f85e0,0x4ff9,0x1068,0xab,0x91,0x08,0x00,0x2b,0x27,0xb3,0xd9}
```

Name/PropId	Datatype	Description
System.Title 0x00000002	VT_LPWSTR	The title of the document.
System.Subject 0x00000003	VT_LPWSTR	The subject of the document.

Name/PropId	Datatype	Description
System.Author 0x00000004	VT_LPWSTR	The author of the document.
System.Keywords 0x00000005	VT_LPWSTR	The document keywords.
System.Comment 0x00000006	VT_LPWSTR	Comments about the document.
DocTemplate 0x00000007	VT_LPWSTR	The name of the template for the document.
System.Document.LastAuthor 0x00000008	VT_LPWSTR	The most recent user who edited document.
System.Document.RevisionNumber 0x00000009	VT_LPWSTR	The current version number of the document.
System.Document.DateCreated 0x0000000A	VT_FILETIME	The total time spent editing the document.
System.Document.DatePrinted 0x0000000B	VT_FILETIME	The date the document was last printed.
System.Document.DateCreated 0x0000000C	VT_FILETIME	The date the document was created.
System.Document.DateSaved 0x0000000D	VT_FILETIME	The date the document was last saved.
System.Document.PageCount 0x0000000E	VT_I4	The number of pages in the document.
System.Document.WordCount 0x0000000F	VT_I4	The number of words in the document.
System.Document.CharacterCount 0x00000010	VT_I4	The number of characters in the document.
DocThumbnail 0x00000011	VT_CF	A thumbnail of the document in clipboard format.
System.ApplicationName 0x00000012	VT_LPWSTR	The name of the application that created the file.

Property Sets for Documents

```
#define  DocPropSetGuid2     \
{0xd5cdd502,0x2e9c,0x101b,0x93,0x97,0x08,0x00,0x2b,0x2c,0xf9,0xae}
```

Name/PropId	Datatype	Description
System.category 0x00000002	VT_LPSTR	The type of document, such as a memo, schedule, or white paper.

Name/PropId	Datatype	Description
System.Document.PresentationFormat 0x00000003	VT_LPSTR	The target format (for example, 35mm, printer, or video) for a presentation in PowerPoint.
System.Document.ByteCount 0x00000004	VT_I4	The number of bytes in a document.
System.Document.LineCount 0x00000005	VT_I4	The number of lines contained in a document.
System.Document.ParagraphCount 0x00000006	VT_I4	The number of paragraphs in a document.
System.Document.SlideCount 0x00000007	VT_I4	The number of slides in a PowerPoint document.
DocNoteCount 0x00000008	VT_I4	The number of pages with notes in a PowerPoint document.
System.Document.HiddenSlideCount 0x00000009	VT_I4	The number of hidden slides in a PowerPoint document.
DocPartTitles 0x0000000D	VT_LPWSTR|VT_VECTOR	The names of document parts. For example, in Excel, part titles are the names of spreadsheets, in PowerPoint slide titles, and in Office Word for Windows, the names of the documents in the master document.
System.Document.Manager 0x0000000E	VT_LPSTR	The name of the manager of the document's author.
System.Company 0x0000000F	VT_LPSTR	The name of the company for which the document was written.

Document characterization

```
#define  DocCharacterGuid     \
{0x560c36c0,0x503a,0x11cf,0xba,0xa1,0x00,0x00,0x4c,0x75,0x2a,0x9a}
```

The name is System.Search.Autosummary, the PropId is 0x00000002, the datatype is VT_LPWSTR, and it is a characterization or abstract of the document.

Music Property Set

```
#define PSGUID_MUSIC     \
{56A3372E-CE9C-11d2-9F0E-006097C686F6}
```

Name/PropId	Datatype	Description
System.Music.Artist 0x00000002	VT_LPWSTR	The artist who recorded the song.
System.Music.Album 0x00000004	VT_LPWSTR	The album on which the song was released.

Name/PropId	Datatype	Description
System.Media.Year 0x00000005	VT_LPWSTR	The year that the song was published.
System.Music.TrackNumber0x00000007	VT_UI4	The track number for the song.
System.Music.Genre0x0000000B	VT_LPWSTR	The song's genre.

Digital Rights Management

```
#define PSGUID_DRM     \
{AEAC19E4-89AE-4508-B9B7-BB867ABEE2ED}
```

This property set contains properties that describe the digital rights associated with some media.

Name/PropId	Datatype	Description
System.DRM.IsProtected 0x00000002	VT_BOOL	TRUE if there is a license.
DrmDescription 0x00000003	VT_LPWSTR	The description of the license.
DrmPlayCount 0x00000004	VT_UI4	The number of times the item can be played.
DrmPlayStarts 0x00000005	VT_FILETIME	The date the play rights start.
DrmPlayExpires 0x00000006	VT_FILETIME	The date the rights expire.

Image Property Set

```
#define PSGUID_IMAGESUMMARYINFORMATION     \
{0x6444048f,0x4c8b,0x11d1,0x8b,0x70,0x8,0x00,0x36,0xb1,0x1a,0x03}
```

Name/PropId	Datatype	Description
ImageFileType 0x00000002	VT_LPWSTR	The type of image file.
System.Image.HorizontalSize 0x00000003	VT_UI4	The horizontal size, in pixels.
System.Image.VerticalSize 0x00000004	VT_UI4	The vertical size, in pixels.
System.Image.HorizontalResolution 0x00000005	VT_UI4	The horizontal resolution, in pixels per inch.
System.Image.VerticalResolution 0x00000006	VT_UI4	The vertical resolution, in pixels per inch.
System.Image.BitDepth	VT_UI4	The number of bits per pixel.

Name/PropId	Datatype	Description
0x00000007		
ImageColorSpace 0x00000008	VT_LPWSTR	The description of the image color space.
ImageCompression 0x00000009	VT_LPWSTR	The description of the image compression.
ImageTransparency 0x0000000A	VT_UI4	The degree of transparency from 0-100.
ImageGammaValue 0x0000000B	VT_UI4	The gamma correction value.
System.Media.FrameCount 0x0000000C	VT_UI4	The frame count for the image.
System.Image.Dimensions 0x0000000D	VT_LPWSTR	The description of the image dimensions.

Audio Property Set

```
#define PSGUID_AUDIO     \
{64440490-4C8B-11D1-8B70-080036B11A03}
```

Audio-Related Properties

Name/PropId	Datatype	Description
AudioFormat 0x00000002	VT_LPWSTR	The audio format.
System.Media.Duration 0x00000003	VT_UI8	The duration, in 100-ns units.
System.Audio.EncodingBitrate 0x00000004	VT_UI4	The average encoding rate, in bits per second.
System.Audio.SampleRate 0x00000005	VT_UI4	The sample rate, in samples per second.
System.Audio.SampleSize 0x00000006	VT_UI4	The sample size, in bits per sample.
System.Audio.ChannelCount 0x00000007	VT_UI4	The number of channels of audio.

Video Property Set

```
#define PSGUID_VIDEO     \
{64440491-4C8B-11D1-8B70-080036B11A03}
```

Name/PropId	Datatype	Description
System.Video.StreamName 0x00000002	VT_LPWSTR	The name of the stream.
System.Video.FrameWidth 0x00000003	VT_UI4	The width, in pixels, of a frame.
System.Video.FrameHeight 0x00000004	VT_UI4	The height, in pixels, of a frame.
VideoTimeLength 0x00000005	VT_UI4	The duration, in 100-ns units.
System.Video.FrameRate 0x00000006	VT_UI4	The number of frames in video.
VideoFrameCount 0x00000007	VT_UI4	The frames per second.
System.Video.EncodingBitrate 0x00000008	VT_UI4	The bits per second.
System.Video.SampleSize 0x00000009	VT_UI4	The bits per sample.
System.Video.Compression 0x0000000A	VT_LPWSTR	The description of video compression.

Mime Properties

```
#define NNTPGuid     \
{0xAA568EEC,0xE0E5,0x11CF,0x8F,0xDA,0x00,0xAA,0x00,0xA1,0x4F,0x93}
```

Name/PropId	Datatype	Description
MsgNewsgroup 0x00000002	VT_LPWSTR	The newsgroup for the message.
MsgSubject 0x00000005	VT_LPWSTR	The subject of the message.
MsgFrom 0x00000006	VT_LPWSTR	Who sent the message.
MsgMessageID 0x00000007	VT_LPWSTR	The unique ID for email.
MsgDate 0x0000000C	VT_FILETIME	When the message was sent.
MsgReceivedDate 0x00000035	VT_FILETIME	When the message was received.
MsgArticleID 0x0000003C	VT_UI4	The unique identifier for the newsgroup article.

Full property table

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.AcquisitionID	{65A98875-3C80-40AB-ABBC-EFDAF77DBEE2}	100	FALSE	TRUE		Int32	4		Hash to determine acquisition session.
System.ApplicationName	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	18	TRUE	TRUE		String	512		
System.Audio.ChannelCount	{64440490-4C8B-11D1-8B70-080036B11A03}	7	FALSE	TRUE		UInt32	4		Indicates the channel count for the audio file. Values: 1 (mono), 2 (stereo).
System.Audio.EncodingBitrate	{64440490-4C8B-11D1-8B70-080036B11A03}	4	FALSE	TRUE		UInt32	4		Indicates the average data rate in Hz for the audio file in "bits per second".
System.Audio.PeakValue	{2579E5D0-1116-4084-BD9A-9B4F7CB4DF5E}	100	FALSE	TRUE		UInt32	4		
System.Audio.SampleRate	{64440490-4C8B-11D1-8B70-080036B11A03}	5	FALSE	TRUE		UInt32	4		Indicates the audio sample rate for the audio file in "samples per second".

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Audio.SampleSize	{64440490-4C8B-11D1-8B70-080036B11A03}	6	FALSE	TRUE		UInt32	4		Indicates the audio sample size for the audio file in "bits per sample".
System.Author	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	4	TRUE	TRUE		String	256	TRUE	
System.Calendar.Duration	{293CA35A-09AA-4DD2-B180-1FE245728A52}	100	TRUE	TRUE		String	512		The duration as specified in a string.
System.Calendar.IsOnline	{BFEE9149-E3E2-49A7-A862-C05988145CEC}	100	FALSE	TRUE		Boolean	2		Indicates whether the event is an online event.
System.Calendar.IsRecurring	{315B9C8D-80A9-4EF9-AE16-8E746DA51D70}	100	FALSE	TRUE		Boolean	2		
System.Calendar.Location	{F6272D18-CECC-40B1-B26A-3911717AA7BD}	100	TRUE	TRUE		String	512		
System.Calendar.OptionalAttendee	{D55BAE5A-	100	TRUE	TRUE		String	256	TRUE	

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
Addresses	3892-417A-A649-C6AC5AAAEAB3}								
System.Calendar.OptionalAttendeeNames	{09429607-582D-437F-84C3-DE93A2B24C3C}	100	TRUE	TRUE		String	256	TRUE	
System.Calendar.OrganizerAddress	{744C8242-4DF5-456C-AB9E-014EFB9021E3}	100	TRUE	TRUE		String	256		Address of the organizer organizing the event.
System.Calendar.OrganizerName	{AAA660F9-9865-458E-B484-01BC7FE3973E}	100	TRUE	TRUE		String	256		Name of the organizer organizing the event.
System.Calendar.ReminderTime	{72FC5BA4-24F9-4011-9F3F-ADD27AFAD818}	100	FALSE	TRUE		DateTime	8		
System.Calendar.RequiredAttendeeAddresses	{0BA7D6C3-568D-4159-AB91-781A91FB71E5}	100	TRUE	TRUE		String	256	TRUE	
System.Calendar.RequiredAttendeeNames	{B33AF30B-F552-	100	TRUE	TRUE		String	256	TRUE	

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	4584-936C-CB93E5CDA29F}								
System.Calendar.Resources	{00F58A38-C54B-4C40-8696-97235980EAE1}	100	TRUE	TRUE		String	512	TRUE	
System.Calendar.ResponseStatus	{188C1F91-3C40-4132-9EC5-D8B03B72A8A2}	100	FALSE	TRUE		UInt16	2		This property stores the status of the user responses to meetings in the user's calendar.
System.Calendar.ShowTimeAs	{5BF396D4-5EB2-466F-BDE9-2FB3F2361D6E}	100	FALSE	TRUE		UInt16	2		
System.Calendar.ShowTimeAsText	{53DA57CF-62C0-45C4-81DE-7610BCEFD7F5}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Calendar.ShowTimeAs. Not intended to be parsed programmatically.
System.Category	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	2	TRUE	TRUE		String	512	TRUE	
System.Comment	{F29F85E0-4FF9-1068-	6	TRUE	TRUE		String	2048		Comments.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	AB91-08002B27B3D9}								
System.Communication.AccountName	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	9	TRUE	TRUE		String	512		Account Name
System.Communication.DateItemExpires	{428040AC-A177-4C8A-9760-F6F761227F9A}	100	FALSE	TRUE		DateTime	8		Date the item expires due to the retention policy.
System.Communication.FollowupIconIndex	{83A6347E-6FE4-4F40-BA9C-C4865240D1F4}	100	FALSE	TRUE		Int32	4		This is the icon index used on messages marked for follow up.
System.Communication.HeaderItem	{C9C34F84-2241-4401-B607-BD20ED75AE7F}	100	FALSE	TRUE		Boolean	2		This property will be true if the item is a header item which means the item hasn't been fully downloaded.
System.Communication.PolicyTag	{EC0B4191-AB0B-4C66-90B6-C6637CDEBBAB}	100	TRUE	TRUE		String	512		This a string used to identify the retention policy applied to the item.
System.Communication.SecurityFlags	{8619A4B6-9F4D-4429-8C0F-B996CA59E3	100	FALSE	TRUE		Int32	4		Security flags associated with the item to know if the item is encrypted, signed or DRM enabled.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	35}								
System.Communication.TaskStatus	{BE1A72C6-9A1D-46B7-AFE7-AFAF8CEF4999}	100	FALSE	TRUE		UInt16	2		
System.Communication.TaskStatusText	{A6744477-C237-475B-A075-54F349829 2A}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Communication.TaskStatus. Not intended to be parsed programmatically.
System.Company	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	15	TRUE	TRUE		String	512		The company or publisher.
System.ComputerName	{28636AA6-953D-11D2-B5D6-00C04FD918D0}	5	FALSE	TRUE		String	512		
System.Contact.Anniversary	{9AD5BADB-CEA7-4470-A03D-B84E51B9949E}	100	FALSE	TRUE		DateTime	8		
System.Contact.AssistantName	{CD102C9C-5540-4A88-A6F6-64E4981C8C	100	TRUE	TRUE		String	256		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	D1}								
System.Contact.AssistantTelephone	{9A93244D-A7AD-4FF8-9B99-45EE4CC09AF6}	100	TRUE	TRUE		String	512		
System.Contact.Birthday	{176DC63C-2688-4E89-8143-A347800F25E9}	47	FALSE	TRUE		DateTime	8		
System.Contact.BusinessAddress	{730FB6DD-CF7C-426B-A03F-BD166CC9EE24}	100	TRUE	TRUE		String	512		
System.Contact.BusinessAddressCity	{402B5934-EC5A-48C3-93E6-85E86A2D934E}	100	TRUE	TRUE		String	512		
System.Contact.BusinessAddressCountry	{B0B87314-FCF6-4FEB-8DFF-A50DA6AF561C}	100	TRUE	TRUE		String	512		
System.Contact.BusinessAddressPostalCode	{E1D4A09E-D758-4CD1-B6EC-34A8B5A73F	100	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	80}								
System.Contact.BusinessAddressPostOfficeBox	{BC4E71CE-17F9-48D5-BEE9-021DF0EA5409}	100	TRUE	TRUE		String	512		
System.Contact.BusinessAddressState	{446F787F-10C4-41CB-A6C4-4D0343551597}	100	TRUE	TRUE		String	512		
System.Contact.BusinessAddressStreet	{DDD1460F-C0BF-4553-8CE4-10433C908FB0}	100	TRUE	TRUE		String	512		
System.Contact.BusinessFaxNumber	{91EFF6F3-2E27-42CA-933E-7C999FBE310B}	100	TRUE	TRUE		String	512		Business fax number of the contact.
System.Contact.BusinessHomePage	{56310920-2491-4919-99CE-EADB06FAFDB2}	100	TRUE	TRUE		String	512		
System.Contact.BusinessTelephone	{6A15E5A0-0A1E-4CD7-BB8C-D2F1B0C929BC}	100	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Contact.CallbackTelephone	{BF53D1C3-49E0-4F7F-8567-5A821D8AC542}	100	TRUE	TRUE		String	512		
System.Contact.CarTelephone	{8FDC6DEA-B929-412B-BA90-397A257465FE}	100	TRUE	TRUE		String	512		
System.Contact.Children	{D4729704-8EF1-43EF-9024-2BD381187FD5}	100	TRUE	TRUE		String	512	TRUE	
System.Contact.CompanyMainTelephone	{8589E481-6040-473D-B171-7FA89C2708ED}	100	TRUE	TRUE		String	512		
System.Contact.Department	{FC9F7306-FF8F-4D49-9FB6-3FFE5C0951EC}	100	TRUE	TRUE		String	512		
System.Contact.EmailAddress	{F8FA7FA3-D12B-4785-8A4E-691A94F7A3E7}	100	TRUE	TRUE		String	256		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Contact.EmailAddress2	{38965063-EDC8-4268-8491-B7723172CF29}	100	TRUE	TRUE		String	256		
System.Contact.EmailAddress3	{644D37B4-E1B3-4BAD-B099-7E7C04966ACA}	100	TRUE	TRUE		String	256		
System.Contact.EmailAddresses	{84D8F337-981D-44B3-9615-C7596DBA17E3}	100	TRUE	TRUE		String	256	TRUE	
System.Contact.EmailName	{CC6F4F24-6083-4BD4-8754-674D0DE87AB8}	100	TRUE	TRUE		String	256		
System.Contact.FileAsName	{F1A24AA7-9CA7-40F6-89EC-97DEF9FFE8DB}	100	TRUE	TRUE		String	256		
System.Contact.FirstName	{14977844-6B49-4AAD-A714-A4513BF60460}	100	TRUE	TRUE		String	256		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Contact.FullName	{635E9051-50A5-4BA2-B9DB-4ED056C77296}	100	TRUE	TRUE		String	256		
System.Contact.Gender	{3C8CEE58-D4F0-4CF9-B756-4E5D24447BCD}	100	TRUE	TRUE		String	512		
System.Contact.GenderValue	{3C8CEE58-D4F0-4CF9-B756-4E5D24447BCD}	101	FALSE	TRUE		UInt16	2		
System.Contact.Hobbies	{5DC2253F-5E11-4ADF-9CFE-910DD01E3E70}	100	TRUE	TRUE		String	512	TRUE	
System.Contact.HomeAddress	{98F98354-617A-46B8-8560-5B1B64BF1F89}	100	TRUE	TRUE		String	512		
System.Contact.HomeAddressCity	{176DC63C-2688-4E89-8143-A347800F25E9}	65	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Contact.HomeAddressCountry	{08A65AA1-F4C9-43DD-9DDF-A33D8E7EAD85}	100	TRUE	TRUE		String	512		
System.Contact.HomeAddressPostalCode	{8AFCC170-8A46-4B53-9EEE-90BAE7151E62}	100	TRUE	TRUE		String	512		
System.Contact.HomeAddressPostOfficeBox	{7B9F6399-0A3F-4B12-89BD-4ADC51C918AF}	100	TRUE	TRUE		String	512		
System.Contact.HomeAddressState	{C89A23D0-7D6D-4EB8-87D4-776A82D493E5}	100	TRUE	TRUE		String	512		
System.Contact.HomeAddressStreet	{0ADEF160-DB3F-4308-9A21-06237B16FA2A}	100	TRUE	TRUE		String	512		
System.Contact.HomeFaxNumber	{660E04D6-81AB-4977-A09F-82313113AB26}	100	TRUE	TRUE		String	512		
System.Contact.	{176DC63	20	TRUE	TR		Str	51		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
HomeTelephone	C-2688-4E89-8143-A347800F25E9}			UE		ing	2		
System.Contact.IMAddress	{D68DBD8A-3374-4B81-9972-3EC30682DB3D}	100	TRUE	TRUE		String	256	TRUE	
System.Contact.JA.CompanyNamePhonetic	{897B3694-FE9E-43E6-8066-260F590C0100}	2	TRUE	TRUE		String	256		
System.Contact.JA.FirstNamePhonetic	{897B3694-FE9E-43E6-8066-260F590C0100}	3	TRUE	TRUE		String	512		
System.Contact.JA.LastNamePhonetic	{897B3694-FE9E-43E6-8066-260F590C0100}	4	TRUE	TRUE		String	256		
System.Contact.JobTitle	{176DC63C-2688-4E89-8143-A347800F25E9}	6	TRUE	TRUE		String	512		
System.Contact.Label	{97B0AD89-	100	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	DF49-49CC-834E-660974FD755B}								
System.Contact.LastName	{8F367200-C270-457C-B1D4-E07C5BCD90C7}	100	TRUE	TRUE		String	256		
System.Contact.MailingAddress	{C0AC206A-827E-4650-95AE-77E2BB74FCC9}	100	TRUE	TRUE		String	512		
System.Contact.MiddleName	{176DC63C-2688-4E89-8143-A347800F25E9}	71	TRUE	TRUE		String	256		
System.Contact.MobileTelephone	{176DC63C-2688-4E89-8143-A347800F25E9}	35	TRUE	TRUE		String	512		
System.Contact.NickName	{176DC63C-2688-4E89-8143-A347800F25E9}	74	TRUE	TRUE		String	256		
System.Contact.	{176	7	TRUE	TR		Str	51		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
OfficeLocation	DC63C-2688-4E89-8143-A347800F25E9}			UE		ing	2		
System.Contact.OtherAddress	{508161FA-313B-43D5-83A1-C1ACCF68622C}	100	TRUE	TRUE		String	512		
System.Contact.OtherAddressCity	{6E682923-7F7B-4F0C-A337-CFCA296687BF}	100	TRUE	TRUE		String	512		
System.Contact.OtherAddressCountry	{8F167568-0AAE-4322-8ED9-6055B7B0E398}	100	TRUE	TRUE		String	512		
System.Contact.OtherAddressPostalCode	{95C656C1-2ABF-4148-9ED3-9EC602E3B7CD}	100	TRUE	TRUE		String	512		
System.Contact.OtherAddressPostOfficeBox	{8B26EA41-058F-43F6-AECC-4035681CE977}	100	TRUE	TRUE		String	512		
System.Contact.OtherAddressState	{71B377D6-E570-	100	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	425F-A170-809FAE73E54E}								
System.Contact.OtherAddressStreet	{FF962609-B7D6-4999-862D-95180D529AEA}	100	TRUE	TRUE		String	512		
System.Contact.PagerTelephone	{D6304E01-F8F5-4F45-8B15-D024A6296789}	100	TRUE	TRUE		String	512		
System.Contact.PersonalTitle	{176DC63C-2688-4E89-8143-A347800F25E9}	69	TRUE	TRUE		String	512		
System.Contact.PrimaryAddressCity	{C8EA94F0-A9E3-4969-A94B-9C62A95324E0}	100	TRUE	TRUE		String	512		
System.Contact.PrimaryAddressCountry	{E53D799D-0F3F-466E-B2FF-74634A3CB7A4}	100	TRUE	TRUE		String	512		
System.Contact.PrimaryAddressPostalCode	{18BBD425-ECFD-	100	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	46EF-B612-7B4A6034EDA0}								
System.Contact.PrimaryAddressPostOfficeBox	{DE5EF3C7-46E1-484E-9999-62C5308394C1}	100	TRUE	TRUE		String	512		
System.Contact.PrimaryAddressState	{F1176DFE-7138-4640-8B4C-AE375DC70A6D}	100	TRUE	TRUE		String	512		
System.Contact.PrimaryAddressStreet	{63C25B20-96BE-488F-8788-C09C407AD812}	100	TRUE	TRUE		String	512		
System.Contact.PrimaryEmailAddress	{176DC63C-2688-4E89-8143-A347800F25E9}	48	TRUE	TRUE		String	256		
System.Contact.PrimaryTelephone	{176DC63C-2688-4E89-8143-A347800F25E9}	25	TRUE	TRUE		String	512		
System.Contact.Profession	{7268AF55-1CE4-4F6E-	100	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	A41F-B6E4EF10E4A9}								
System.Contact.SpouseName	{9D2408B6-3167-422B-82B0-F583B7A7CFE3}	100	TRUE	TRUE		String	256		
System.Contact.Suffix	{176DC63C-2688-4E89-8143-A347800F25E9}	73	TRUE	TRUE		String	512		
System.Contact.TelexNumber	{C554493C-C1F7-40C1-A76C-EF8C0614003E}	100	TRUE	TRUE		String	512		
System.Contact.TTYTDDTelephone	{AAF16BAC-2B55-45E6-9F6D-415EB94910DF}	100	TRUE	TRUE		String	512		
System.Contact.WebPage	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	18	TRUE	TRUE		String	4168		
System.ContentStatus	{D5CDD502-2E9C-101B-	27	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	9397-08002B2CF9AE}								
System.ContentType	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	26	TRUE	TRUE		String	512		
System.ContentUrl	{49691C90-7E17-101A-A91C-08002B2ECDA9}	10	TRUE	TRUE		String	4168		In the Open Search Provider, an item is usually made up of a link and some content. This represents the URL to the content.
System.Copyright	{64440492-4C8B-11D1-8B70-080036B11A03}	11	TRUE	TRUE		String	512		
System.DateAccessed	{B725F130-47EF-101A-A5F1-02608C9EEBAC}	16	FALSE	TRUE		DateTime	8		
System.DateAcquired	{2CBAA8F5-D81F-47CA-B17A-F8D8230013 1}	100	FALSE	TRUE		DateTime	8		The date the file entered the system via acquisition. This is not the same as System.DateImported. This would apply, for example, to transfer an image from a camera or to music purchase from an online site.
System.DateArchived	{43F8D7B7-A444-4F87-9383-	100	FALSE	TRUE		DateTime	8		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	52271C9B915C}								
System.DateCompleted	{72FAB781-ACDA-43E5-B155-B2434F85E678}	100	FALSE	TRUE		DateTime	8		
System.DateCreated	{B725F130-47EF-101A-A5F1-02608C9EEBAC}	15	FALSE	TRUE		DateTime	8		The date and time the item was created. The WSS friendly name is 'create'.
System.DateImported	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	18258	FALSE	TRUE		DateTime	8		The date the file is imported into a separate database. This is not the same as System.DateAcquired. (For example, 2003:05:22 13:55:04)
System.DateModified	{B725F130-47EF-101A-A5F1-02608C9EEBAC}	14	FALSE	TRUE		DateTime	8		The date and time of the last write to the item. The WSS friendly name is 'write'.
System.Document.ByteCount	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	4	FALSE	TRUE		Int32	4		
System.Document.CharacterCount	{F29F85E0-4FF9-1068-AB91-08002B27B3	16	FALSE	TRUE		Int32	4		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	D9}								
System.Document.ClientID	{276D7BB0-5B34-4FB0-AA4B-158ED12A1809}	100	TRUE	TRUE		String	512		
System.Document.Contributor	{F334115E-DA1B-4509-9B3D-1195044DC7ABB}	100	TRUE	TRUE		String	512	TRUE	
System.Document.DateCreated	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	12	FALSE	TRUE		DateTime	8		This property is stored in the document, not obtained from the file system.
System.Document.DatePrinted	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	11	FALSE	TRUE		DateTime	8		
System.Document.DateSaved	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	13	FALSE	TRUE		DateTime	8		
System.Document.Division	{1E005EE6-BF27-428B-B01C-79676ACD2870}	100	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Document.DocumentID	{E08805C8-E395-40DF-80D2-54F0D6C43154}	100	TRUE	TRUE		String	512		
System.Document.HiddenSlideCount	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	9	FALSE	TRUE		Int32	4		
System.Document.LastAuthor	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	8	TRUE	TRUE		String	256		
System.Document.LineCount	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	5	FALSE	TRUE		Int32	4		
System.Document.Manager	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	14	TRUE	TRUE		String	512		
System.Document.PageCount	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	14	FALSE	TRUE		Int32	4		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Document.ParagraphCount	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	6	FALSE	TRUE		Int32	4		
System.Document.PresentationFormat	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	3	TRUE	TRUE		String	512		
System.Document.RevisionNumber	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	9	TRUE	TRUE		String	512		
System.Document.SlideCount	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	7	FALSE	TRUE		Int32	4		
System.Document.TotalEditingTime	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	10	FALSE	TRUE		UInt64	8		100ns units, not milliseconds. VT_FILETIME for IPropertySetStorage handlers (legacy)
System.Document.Version	{D5CDD502-2E9C-101B-9397-08002B2CF9	29	FALSE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	AE}								
System.Document.WordCount	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	15	FALSE	TRUE		Int32	4		
System.DRM.IsProtected	{AEAC19E4-89AE-4508-B9B7-BB867ABEE2ED}	2	FALSE	TRUE		Boolean	2		
System.DueDate	{3F8472B5-E0AF-4DB2-8071-C53FE76AE7CE}	100	FALSE	TRUE		DateTime	8		
System.EndDate	{C75FAA05-96FD-49E7-9CB4-9F601082D553}	100	FALSE	TRUE		DateTime	8		
System.FileAttributes	{B725F130-47EF-101A-A5F1-02608C9EEBAC}	13	FALSE	TRUE		UInt32	4		This is the WIN32_FIND_DATA dwFileAttributes for the file-based item.
System.FileDescription	{0CEF7D53-FA64-11D1-A203-0000F81FEDEE}	3	TRUE	FALSE		String	512		This is a user-friendly description of the file.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.FileExtension	{E4F10A3C-49E6-405D-8288-A23BD4EEAA6C}	100	TRUE	TRUE		String	512		This is the file extension of the file-based item, including the leading period. If System.FileName is VT_EMPTY, then this property is too. Otherwise, it is to be derived appropriately by the data source from System.FileName. If System.FileName does not have a file extension, this value is VT_EMPTY. To obtain the type of any item (including an item that is not a file), use System.ItemType.Example values: If the path is... The property value is... ---------- ---------- "c:\folder\bar\hello.txt" ".txt" "\\server\share\mydir\goodnews.doc" ".doc" "\\server\share\numbers.xls" ".xls" "\\server\share\folder" VT_EMPTY "c:\folder\MyFolder" VT_EMPTY [desktop] VT_EMPTY
System.FileFRN	{B725F130-47EF-101A-A5F1-02608C9EEBAC}	21	FALSE	TRUE		UInt64	8		This is the unique file ID, also known as the File Reference Number. For a given file, this is the same value as is found in the structure variable FILE_ID_BOTH_DIR_INFO.FileId, via GetFileInformationByHandleEx().
System.FileName	{41CF5AE0-F75A-4806-BD87-59C7D9248EB9}	100	TRUE	TRUE		String	520		This is the file name (including extension) of the file. It is possible that the item might not exist on a filesystem (that is, it cannot be opened using CreateFile). Nonetheless, if the item is represented as a file from the logical sense (and its name follows standard Win32 file-naming syntax), then the data source has to emit this property. If an item is not a file, then the value for this property is VT_EMPTY. SeeSystem.ItemNameDisplay. This has the same value as System.ParsingName for items that are provided by the Shell's file folder. Example values: if the

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
									path is... The property value is...---------- ---------- -----"c:\folder\bar\hello.txt" "hello.txt""\\server\share\mydir\goodnews.doc" "goodnews.doc""\\server\share\numbers.xls" "numbers.xls""c:\folder\MyFolder" "MyFolder"(email message) VT_EMPTY(song on portable device) "song.wma"
System.FileOwner	{9B174B34-40FF-11D2-A27E-00C04FC30871}	4	TRUE	TRUE		String	256		This is the owner of the file, according to the file system.
System.FlagColor	{67DF94DE-0CA7-4D6F-B792-053A3E4F03CF}	100	FALSE	TRUE		UInt16	2		name="Purple" value="1" text="Purple"name="Orange" value="2" text="Orange"name="Green" value="3" text="Green" name="Yellow" value="4" text="Yellow"name="Blue" value="5" text="Blue" name="Red" value="6" text="Red"
System.FlagColorText	{45EAE747-8E2A-40AE-8CBF-CA52ABA6152A}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.FlagColor. Not intended to be parsed programmatically.
System.FlagStatus	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	12	FALSE	TRUE		Int32	4		Status of Flag. Values: (0=none 1=white 2=Red).
System.FlagStatusText	{DC54FD2E-189D-4871-	100	TRUE	TRUE		String	512		This is the user-friendly form of System.FlagStatus. Not intended to be parsed programmatically.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	AA01-08C2F57A4ABC}								
System.GPS.Date	{3602C812-0F3B-45F0-85AD-603468D69423}	100	FALSE	TRUE		DateTime	8		Date and time of the GPS record.
System.IconIndex	{5CBF2787-48CF-4208-B90E-EE5E5D420294}	26	FALSE	TRUE		Int32	4		
System.Identity	{A26F4AFC-7346-4299-BE47-EB1AE613139F}	100	TRUE	TRUE		String	512		
System.Image.BitDepth	{6444048F-4C8B-11D1-8B70-080036B11A03}	7	FALSE	TRUE		UInt32	4		
System.Image.Compression	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	259	FALSE	TRUE		UInt16	2		Indicates the image compression level. PropertyTagCompression.
System.Image.CompressionText	{3F08E66F-2F44-4BB9-A682-AC35D256	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Image.Compression. Not intended to be parsed programmatically.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	2322}								
System.Image.Dimensions	{6444048F-4C8B-11D1-8B70-080036B11A03}	13	TRUE	TRUE		String	512		Indicates the dimensions of the image.
System.Image.HorizontalResolution	{6444048F-4C8B-11D1-8B70-080036B11A03}	5	FALSE	TRUE		Double	8		
System.Image.HorizontalSize	{6444048F-4C8B-11D1-8B70-080036B11A03}	3	FALSE	TRUE		UInt32	4		
System.Image.VerticalResolution	{6444048F-4C8B-11D1-8B70-080036B11A03}	6	FALSE	TRUE		Double	8		
System.Image.VerticalSize	{6444048F-4C8B-11D1-8B70-080036B11A03}	4	FALSE	TRUE		UInt32	4		
System.Importance	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	11	FALSE	TRUE		Int32	4		
System.Importan	{A3B2	10	TRUE	TR		Str	51		This is the user-friendly form of

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
ceText	9791-7713-4E1D-BB40-17DB85F01831}	0		UE		ing	2		System.Importance. Not intended to be parsed programmatically.
System.IsAttachment	{F23F425C-71A1-4FA8-922F-678EA4A60408}	100	FALSE	TRUE		Boolean	2		Identifies this item as an attachment.
System.IsDeleted	{5CDA5FC8-33EE-4FF3-9094-AE7BD8868C4D}	100	FALSE	TRUE		Boolean	2		
System.IsEncrypted	{90E5E14E-648B-4826-B2AA-ACAF790E3513}	10	FALSE	TRUE		Boolean	2		Holds a value indicating whether the item encrypted?
System.IsFlagged	{5DA84765-E3FF-4278-86B0-A27967FBDD03}	100	FALSE	TRUE		Boolean	2		
System.IsFlaggedComplete	{A6F360D2-55F9-48DE-B909-620E090A647C}	100	FALSE	TRUE		Boolean	2		
System.IsFolder	{0932	10	FALSE	TR		Bo	2		Set this to true if the item is a

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	9B74-40A3-4C68-BF07-AF9A572F607C}	0		UE		olean			folder.
System.IsIncomplete	{346C8BD1-2E6A-4C45-89A4-61B78E8E700F}	100	FALSE	TRUE		Boolean	2		Indicates whether the message was not completely received for some error condition.
System.IsRead	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	10	FALSE	TRUE		Boolean	2		Has the item been read?
System.ItemAuthors	{D0A04F0A-462A-48A4-BB2F-3706E88DBD7D}	100	TRUE	TRUE		String	256	TRUE	This is the generic list of authors associated with an item. For example, the artist name for a track is the item author.
System.ItemDate	{F7DB74B4-4287-4103-AFBA-F1B13DCD75CF}	100	FALSE	TRUE		DateTime	8		This is the main date for an item. The date of interest. For example, for photos this maps to System.Photo.DateTaken.
System.ItemFolderNameDisplay	{B725F130-47EF-101A-A5F1-02608C9EEBAC}	2	TRUE	TRUE		String	512		This is the user-friendly display name of the parent folder of an item. If System.ItemFolderPathDisplay is VT_EMPTY, then this property is too. Otherwise, it is derived appropriately by the data source from System.ItemFolderPathDisplay. If the folder is a file folder, the value will be localized if a localized name is available

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
									.Example values: If the path is... The property value is...---------- ---------- ----------"c:\folder\bar\hello.txt" "bar""\\server\share\mydir\goodnews.doc" "mydir""\\server\share\numbers.xls" "share""c:\folder\MyFolder" "folder""/Mailbox Account/Inbox/'Re: Hello!'" "Inbox"
System.ItemFolderPathDisplay	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	6	TRUE	TRUE		String	520		This is the user-friendly display path of the parent folder of an item.If System.ItemPathDisplay is VT_EMPTY, then this property is too. Otherwise, it is derived appropriately by the data source from System.ItemPathDisplay.Example values:If the path is... The property value is...---------- ---------- ----------"c:\folder\bar\hello.txt" "c:\folder\bar""\\server\share\mydir\goodnews.doc" "\\server\share\mydir""\\server\share\numbers.xls" "\\server\share""c:\folder\MyFolder" "c:\folder""/Mailbox Account/Inbox/'Re: Hello!'" "/Mailbox Account/Inbox"
System.ItemFolderPathDisplayNarrow	{DABD30ED-0043-4789-A7F8-D013A4736622}	100	TRUE	TRUE		String	520		This is the user-friendly display path of the parent folder of an item. The format of the string is to be tailored such that the folder name comes first, to optimize for a narrow viewing column. If the folder is a file folder, the value includes localized names if they are present. If System.ItemFolderPathDisplay is VT_EMPTY, then this property is too. Otherwise, it is to be derived appropriately by the data source from System.ItemFolderPathDisplay.Example values: ---------- "c:\folder\bar\hello.txt" "bar (c:\folder)""\\server\share\mydir\goodnews.doc" "mydir (\\server\share)""\\server\share

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
									\numbers.xls" "share (\\server)""c:\folder\MyFolder" "folder (c:\)""/Mailbox Account/Inbox/'Re: Hello!'" "Inbox (/Mailbox Account)"
System.ItemName	{6B8DA074-3B5C-43BC-886F-0A2CDCE00B6F}	100	FALSE	TRUE		String	520		This is the base-name of the System.ItemNameDisplay. If the item is a file this property includes the extension in all cases, and will be localized if a localized name is available. If the item is a message, then the value of this property does not include the forwarding or reply prefixes (see System.ItemNamePrefix).
System.ItemNameDisplay	{B725F130-47EF-101A-A5F1-02608C9EEBAC}	10	TRUE	TRUE		String	520		This is the display name in "most complete" form. This is the best effort unique representation of the name of an item that makes sense for end users to read. It is the concatenation of System.ItemNamePrefix and System.ItemName. If the item is a file this property includes the extension in all cases, and will be localized if a localized name is available. There are acceptable cases when System.FileName is not VT_EMPTY, yet the value of this property is completely different. Email messages are a key example. If the item is an email message, the item name is likely the subject. In that case, the value must be the concatenation of the System.ItemNamePrefix and System.ItemName. Since the value of System.ItemNamePrefix excludes any trailing whitespace, the concatenation must include whitespace when generating System.ItemNameDisplay. Note that this property is not guaranteed to be unique, but the idea is to promote the most likely candidate that can be unique and also makes sense for end users. For example, for documents, you might think about using System.Title as the System.ItemNameDisplay, but in practice the title of the documents might not be useful

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
									or unique enough to be of value as the sole System.ItemNameDisplay. Instead, providing the value of System.FileName as the value of System.ItemNameDisplay is a better candidate. In Windows Mail, the emails are stored in the file system as .eml files and the System.FileName for those files are not human-friendly as they contain GUIDs. In this example, promoting System.Subject as System.ItemNameDisplay makes more sense. Compatibility notes: Shell folder implementations on Vista: use PKEY_ItemNameDisplay for the name column when you want Explorer to call ISF::GetDisplayNameOf(SHGDN_NORMAL) to get the value of the name. Use another PKEY (like PKEY_ItemName) when you want Explorer to call either the folder's property store orISF2::GetDetailsEx in order to get the value of the name. Shell folder implementations on XP: the first column needs to be the name column, and Explorer will call ISF::GetDisplayNameOf to get the value of the name. The PKEY/SCID does not matter. Example values:File: "hello.txt"Message: "Re: Let's talk about Tom's argyle socks!"Device folder: "song.wma"Folder: "Documents"
System.ItemNamePrefix	{D7313FF1-A77A-401C-8C99-3DBDD68ADD36}	100	FALSE	TRUE		String	520		This is the prefix of an item, used for email messages where the subject begins with "Re:" which is the prefix. If the item is a file, then the value of this property is VT_EMPTY. If the item is a message, then the value of this property is the forwarding or reply prefixes (including delimiting colon, but no whitespace), or VT_EMPTY if there is no prefix. Example values: System.ItemNamePrefix System.ItemName System.ItemNameDisplay---------- ----------

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
									----------VT_EMPTY "Great day" "Great day""Re:" "Great day" "Re: Great day""Fwd: " "Monthly budget" "Fwd: Monthly budget"VT_EMPTY "accounts.xls" "accounts.xls"
System.ItemParticipants	{D4D0AA16-9948-41A4-AA85-D97FF9646993}	100	TRUE	TRUE		String	256	TRUE	This is the generic list of people associated with an item and who contributed to the item. For example, this is the combination of people in the To list, Cc list and Sender of an email message.
System.ItemPathDisplay	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	7	TRUE	TRUE		String	520		This is the user-friendly display path to the item. If the item is a file or folder this property includes the extension in all cases, and will be localized if a localized name is available. For other items, this is the user-friendly equivalent, assuming the item exists in hierarchical storage. Unlike System.ItemUrl, this property value does not include the URL scheme. To parse an item path, use System.ItemUrl or System.ParsingPath. To reference shell namespace items using shell APIs, use System.ParsingPath. Example values: If the path is... The property value is...---------- ----------"c:\folder\bar\hello.txt" "c:\folder\bar\hello.txt""\\server\share\mydir\goodnews.doc" "\\server\share\mydir\goodnews.doc""\\server\share\numbers.xls" "\\server\share\numbers.xls""c:\folder\MyFolder" "c:\folder\MyFolder""/Mailbox Account/Inbox/'Re: Hello!'" "/Mailbox Account/Inbox/'Re: Hello!'"
System.ItemPathDisplayNarrow	{28636AA6-953D-11D2-B5D6-	8	FALSE	TRUE		String	520		This is the user-friendly display path to the item. The format of the string is tailored such that the name comes first, to optimize for a narrow viewing

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	00C04FD918D0}								column. If the item is a file, the value excludes the file extension, and includes localized names if they are present. If the item is a message, the value includes the System.ItemNamePrefix. To parse an item path, use System.ItemUrl or System.ParsingPath. Example values: If the path is... The property value is...---------- ---------- "c:\folder\bar\hello.txt" "hello (c:\folder\bar)""\\server\share\mydir\goodnews.doc" "goodnews (\\server\share\mydir)""\\server\share\folder" "folder (\\server\share)""c:\folder\MyFolder" "MyFolder (c:\folder)""/Mailbox Account/Inbox/'Re: Hello!'" "Re: Hello! (/Mailbox Account/Inbox)"
System.ItemType	{28636AA6-953D-11D2-B5D6-00C04FD918D0}	11	TRUE	TRUE		String	512		This is the canonical type of the item and is intended to be programmatically parsed. If there is no canonical type, the value is VT_EMPTY. If the item is a file (that is, System.FileName is not VT_EMPTY), the value is the same as System.FileExtension. Use System.ItemTypeText when you want to display the type to end users in a view. (If the item is a file, passing the System.ItemType value to PSFormatForDisplay will result in the same value as System.ItemTypeText.) Example values: If the path is... The property value is...---------- ---------- "c:\folder\bar\hello.txt" ".txt""\\server\share\mydir\goodnews.doc" ".doc""\\server\share\folder" "Directory""c:\folder\MyFolder" "Directory"[desktop] "Folder""/Mailbox Account/Inbox/'Re: Hello!'" "MAPI/IPM.Message"
System.ItemType	{B725F130-	4	TRUE	TR		Str	51		This is the user friendly type name of the item. This is not

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
Text	47EF-101A-A5F1-02608C9EEBAC}			UE		ing	2		intended to be programmatically parsed. If System.ItemType is VT_EMPTY, the value of this property is also VT_EMPTY. If the item is a file, the value of this property is the same as if you passed thefile's System.ItemType value to PSFormatForDisplay.This property can not be confused with System.Kind, where System.Kind is a high-level user friendly kind name. For example, for a document, System.Kind = "Document" andSystem.Item.Type = ".doc" and System.Item.TypeText = "Microsoft Word Document" Example values: If the path is… The property value is…----------"c:\folder\bar\hello.txt" "Text File""\\server\share\mydir\goodnews.doc" "Microsoft Word Document""\\server\share\folder" "File Folder""c:\folder\MyFolder" "File Folder""/Mailbox Account/Inbox/'Re: Hello!'" "Outlook E-mail Message"
System.ItemUrl	{49691C90-7E17-101A-A91C-08002B2ECDA9}	9	TRUE	TRUE		String	4168		This always represents a well formed URL that points to the item. To reference shell namespace items using shell APIs, use System.ParsingPath. Example values:Files: "file:///c:/folder/bar/hello.txt""csc://{GUID}/…"Messages: "mapi://…"
System.Journal.Contacts	{DEA7C82C-1D89-4A66-9427-A4E3DEBABCB1}	100	TRUE	TRUE		String	512	TRUE	
System.Journal.EntryType	{95BEB1FC-326D-4644-	100	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	B396-CD3ED90E6DDF}								
System.Keywords	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	5	TRUE	TRUE		String	512	TRUE	The keywords for the item. Also referred to as tags.
System.Kind	{1E3EE840-BC2B-476C-8237-2ACD1A839B22}	3	TRUE	TRUE		String	512	TRUE	System.Kind is used to map extensions to various .Search folders. Extensions are mapped to Kinds at HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\KindMapThe list of kinds is not extensible. "Calendar" "Communication" "Contact" "Document" "Email" "Feed" "Folder" "Game" "InstantMessage" "Journal" "Link" "Movie" "Music" "Note" "Picture" "Program" "RecordedTV" "SearchFolder" "Task" "Video" "WebHistory"
System.KindText	{F04BEF95-C585-4197-A2B7-DF46FDC9EE6D}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Kind. Not intended to be parsed programmatically.
System.Language	{D5CDD502-2E9C-101B-9397-08002B2CF9AE}	28	TRUE	TRUE		String	512		
System.Link.TargetExtension	{7A7D76F4-	2	TRUE	FALSE		String	512	TRUE	The file extension of the link target. See System.File.Extension

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	B630-4BD7-95FF-37CC51A975C9}								
System.Link.TargetParsingPath	{B9B4B3FC-2B51-4A42-B5D8-324146AFCF25}	2	FALSE	TRUE		String	520		This is the shell namespace path to the target of the link item. This path can be passed to SHParseDisplayName to parse the path to the correct shell folder. If the target item is a file, the value is identical to System.ItemPathDisplay. If the target item cannot be accessed through the shell namespace, this value is VT_EMPTY.
System.Link.TargetSFGAOFlags	{B9B4B3FC-2B51-4A42-B5D8-324146AFCF25}	8	FALSE	TRUE		UInt32	4		IshellFolder::GetAttributesOf flags for the target of a link, with SFGAO_PKEYSFGAOMASK attributes masked out.
System.Link.TargetSFGAOFlagsStrings	{D69 42081 - D53B-443D-AD47-5E059D9CD27A}	3	TRUE	FALSE		String	512	TRUE	Expresses the SFGAO flags of a link as string values and is used as a query optimization. See PKEY_Shell_SFGAOFlagsStrings for possible values of this.
System.Link.TargetUrl	{5CBF2787-48CF-4208-B90E-EE5E5D420294}	2	TRUE	TRUE		String	4168		
System.Media.AverageLevel	{09EDD5B6-B301-43C5-9990-D00302EFFD46}	100	FALSE	TRUE		UInt32	4		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Media.ClassPrimaryID	{64440492-4C8B-11D1-8B70-080036B11A03}	13	FALSE	TRUE		String	512		
System.Media.ClassSecondaryID	{64440492-4C8B-11D1-8B70-080036B11A03}	14	FALSE	TRUE		String	512		
System.Media.CollectionGroupID	{64440492-4C8B-11D1-8B70-080036B11A03}	24	FALSE	TRUE		String	512		
System.Media.CollectionID	{64440492-4C8B-11D1-8B70-080036B11A03}	25	FALSE	TRUE		String	512		
System.Media.ContentDistributor	{64440492-4C8B-11D1-8B70-080036B11A03}	18	FALSE	TRUE		String	512		
System.Media.ContentID	{64440492-4C8B-11D1-8B70-080036B11A03}	26	FALSE	TRUE		String	512		
System.Media.CreatorApplication	{64440492-4C8B-	27	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	11D1-8B70-080036B11A03}								
System.Media.CreatorApplicationVersion	{64440492-4C8B-11D1-8B70-080036B11A03}	28	TRUE	TRUE		String	512		
System.Media.DateEncoded	{2E4B640D-5019-46D8-8881-55414CC5CAA0}	100	FALSE	TRUE		DateTime	8		DateTime is in UTC (in the doc, not file system).
System.Media.DateReleased	{DE41CC29-6971-4290-B472-F59F2E2F31E2}	100	TRUE	TRUE		String	512		
System.Media.Duration	{64440490-4C8B-11D1-8B70-080036B11A03}	3	FALSE	TRUE		UInt64	8		100ns units, not milliseconds
System.Media.DVDID	{64440492-4C8B-11D1-8B70-080036B11A03}	15	FALSE	TRUE		String	512		
System.Media.EncodedBy	{64440492-4C8B-11D1-8B70-	36	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	080036B11A03}								
System.Media.FrameCount	{6444048F-4C8B-11D1-8B70-080036B11A03}	12	FALSE	TRUE		UInt32	4		Indicates the frame count for the image.
System.Media.MCDI	{64440492-4C8B-11D1-8B70-080036B11A03}	16	FALSE	TRUE		String	512		
System.Media.MetadataContentProvider	{64440492-4C8B-11D1-8B70-080036B11A03}	17	FALSE	TRUE		String	512		
System.Media.Producer	{64440492-4C8B-11D1-8B70-080036B11A03}	22	TRUE	TRUE		String	256	TRUE	
System.Media.ProtectionType	{64440492-4C8B-11D1-8B70-080036B11A03}	38	FALSE	TRUE		String	512		If media is protected, how is it protected?
System.Media.ProviderRating	{64440492-4C8B-11D1-8B70-080036B11A03}	39	FALSE	TRUE		String	512		Rating (0 – 99) supplied by metadata provider

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Media.ProviderStyle	{64440492-4C8B-11D1-8B70-080036B11A03}	40	FALSE	TRUE		String	512		Style of music or video, supplied by metadata provider
System.Media.Publisher	{64440492-4C8B-11D1-8B70-080036B11A03}	30	TRUE	TRUE		String	512		
System.Media.SubscriptionContentId	{9AEBAE7A-9644-487D-A92C-657585ED751A}	100	FALSE	TRUE		String	512		
System.Media.SubTitle	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	38	TRUE	TRUE		String	512		
System.Media.UniqueFileIdentifier	{64440492-4C8B-11D1-8B70-080036B11A03}	35	FALSE	TRUE		String	512		
System.Media.UserNoAutoInfo	{64440492-4C8B-11D1-8B70-080036B11A03}	41	FALSE	TRUE		String	512		If true, do NOT alter this file's metadata. Set by user.
System.Media.UserWebUrl	{64440492-4C8B-	34	FALSE	TRUE		String	4168		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	11D1-8B70-080036B11A03}								
System.Media.Writer	{64440492-4C8B-11D1-8B70-080036B11A03}	23	TRUE	TRUE		String	256	TRUE	
System.Media.Year	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	5	FALSE	TRUE		UInt32	4		
System.Message.AttachmentContents	{3143BF7C-80A8-4854-8880-E2E40189BDD0}	100	TRUE	FALSE		String	512		
System.Message.AttachmentNames	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	21	TRUE	TRUE		String	512	TRUE	The names of the attachments in a message
System.Message.BccAddress	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	2	TRUE	TRUE		String	256	TRUE	Lists the addresses in the Bcc: field
System.Message.BccName	{E3E0584C-B788-4A5A-BB20-7F5A4	3	TRUE	TRUE		String	256	TRUE	Lists the names in the Bcc: field

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	4C9ACDD}								
System.Message.CcAddress	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	4	TRUE	TRUE		String	256	TRUE	Lists the addresses in the Cc: field
System.Message.CcName	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	5	TRUE	TRUE		String	256	TRUE	Indicates the names listed in the Cc: field
System.Message.ConversationID	{DC8F80BD-AF1E-4289-85B6-3DFC1B493992}	100	TRUE	TRUE		String	512		
System.Message.ConversationIndex	{DC8F80BD-AF1E-4289-85B6-3DFC1B493992}	101	FALSE	TRUE		Buffer	1024		
System.Message.DateReceived	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	20	FALSE	TRUE		DateTime	8		Date and Time communication was received.
System.Message.DateSent	{E3E0584C-B788-4A5A-BB20-7F5A44C9A	19	FALSE	TRUE		DateTime	8		Date and Time communication was sent.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	CDD}								
System.Message.Flags	{A82D9EE7-CA67-4312-965E-226BCEA85023}	100	FALSE	TRUE		Int32	4		These are flags associated with email messages to know if a read receipt is pending, etc. The values stored here by Outlook are defined for PR_MESSAGE_FLAGS on MSDN.
System.Message.FromAddress	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	13	TRUE	TRUE		String	256	TRUE	
System.Message.FromName	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	14	TRUE	TRUE		String	256	TRUE	Specifies the address in the from field as a person's name.
System.Message.HasAttachments	{9C1FCF74-2D97-41BA-B4AE-CB2E3661A6E4}	8	FALSE	TRUE		Boolean	2		
System.Message.IsFwdOrReply	{9A9BC088-4F6D-469E-9919-E705412040F9}	100	FALSE	TRUE		Int32	4		
System.Message.MessageClass	{CD9ED458-08CE-418F-A70E-F912C7BB9C5C}	103	TRUE	TRUE		String	512		Describes what type of Outlook message this is (meeting, task, mail, etc.)

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Message.ProofInProgress	{9098F33C-9A7D-48A8-8DE5-2E1227A64E91}	100	FALSE	TRUE		Boolean	2		This property will be true if the message junk email proofing is still in progress.
System.Message.SenderAddress	{0BE1C8E7-1981-4676-AE14-FDD78F05A6E7}	100	TRUE	TRUE		String	256		
System.Message.SenderName	{0DA41CFA-D224-4A18-AE2F-596158DB4B3A}	100	TRUE	TRUE		String	256		
System.Message.Store	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	15	FALSE	TRUE		String	512		The store (aka protocol handler) FILE, MAIL, OUTLOOKEXPRESS
System.Message.ToAddress	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	16	TRUE	TRUE		String	256	TRUE	Addresses in To: field
System.Message.ToDoFlags	{1F856A9F-6900-4ABA-9505-2D5F1B4D66CB}	100	FALSE	TRUE		Int32	4		Flags associated with a message flagged to know if it's still active, if it was custom flagged, etc.
System.Message.	{BCCC8A3	10	TRUE	TR		Str	51		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
ToDoTitle	C-8CEF-42E5-9B1C-C69079398BC7}	0		UE		ing	2		
System.Message.ToName	{E3E0584C-B788-4A5A-BB20-7F5A44C9ACDD}	17	TRUE	TRUE		String	256	TRUE	Person names in To: field
System.MileageInformation	{FDF84370-031A-4ADD-9E91-0D775F1C6605}	100	TRUE	TRUE		String	512		
System.MIMEType	{0B63E350-9CCC-11D0-BCDB-00805FCCCE04}	5	TRUE	TRUE		String	512		The MIME type. Eg, for EML files: 'message/rfc822'.
System.Music.AlbumArtist	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	13	TRUE	TRUE		String	256		
System.Music.AlbumID	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	100	TRUE	TRUE		String	2048		Concatenation of System.Music.AlbumArtist and System.Music.AlbumTitle, suitable for indexing and display. Used to differentiate albums with the same title from different artists.
System.Music.AlbumTitle	{56A3372E-CE9C-	4	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	11D2-9F0E-006097C686F6}								
System.Music.Artist	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	2	TRUE	TRUE		String	256	TRUE	
System.Music.BeatsPerMinute	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	35	TRUE	TRUE		String	512		
System.Music.Composer	{64440492-4C8B-11D1-8B70-080036B11A03}	19	TRUE	TRUE		String	256	TRUE	
System.Music.Conductor	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	36	TRUE	TRUE		String	256	TRUE	
System.Music.ContentGroupDescription	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	33	FALSE	TRUE		String	512		
System.Music.DisplayArtist	{FD122953-FA93-4EF7-92C3-04C94	100	TRUE	TRUE		String	256		This property returns the best representation of Album Artist for a given music file based upon AlbumArtist, ContributingArtist and compilation info.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	6B2F7C8}								
System.Music.Genre	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	11	TRUE	TRUE		String	512	TRUE	
System.Music.InitialKey	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	34	TRUE	TRUE		String	512		
System.Music.IsCompilation	{C449D5CB-9EA4-4809-82E8-AF9D59DED6D1}	100	FALSE	TRUE		Boolean	2		Indicates whether the file is part of a compilation.
System.Music.Lyrics	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	12	TRUE	FALSE		String	512		
System.Music.Mood	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	39	TRUE	TRUE		String	512		
System.Music.PartOfSet	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	37	FALSE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Music.Period	{64440492-4C8B-11D1-8B70-080036B11A03}	31	TRUE	TRUE		String	512		
System.Music.TrackNumber	{56A3372E-CE9C-11D2-9F0E-006097C686F6}	7	FALSE	TRUE		UInt32	4		
System.Note.Color	{4776CAFA-BCE4-4CB1-A23E-265E76D8EB11}	100	FALSE	TRUE		UInt16	2		name="Blue" value="0"name="Green" value="1"name="Pink" value="2"name="Yellow" value="3"name="White" value="4"name="LightGreen" value="5"
System.Note.ColorText	{46B4E8DE-CDB2-440D-885C-1658EB65B914}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Note.Color. Not intended to be parsed programmatically.
System.OriginalFileName	{0CEF7D53-FA64-11D1-A203-0000F81FEDEE}	6	TRUE	TRUE		String	520		
System.ParentalRating	{64440492-4C8B-11D1-8B70-080036B11A03}	21	TRUE	TRUE		String	512		
System.ParentalRatingReason	{10984E0A-F9F2-	100	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	4321-B7EF-BAF195AF4319}								
System.ParsingName	{28636AA6-953D-11D2-B5D6-00C04FD918D0}	24	TRUE	TRUE		String	520		The shell namespace name of an item relative to a parent folder. This name can be passed to IshellFolder::ParseDisplayName() of the parent shell folder.
System.Photo.Aperture	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	37378	FALSE	TRUE		Double	8		PropertyTagExifAperture. Calculated from PKEY_Photo_ApertureNumerator and PKEY_Photo_ApertureDenominator
System.Photo.CameraManufacturer	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	271	TRUE	TRUE		String	512		
System.Photo.CameraModel	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	272	TRUE	TRUE		String	512		
System.Photo.ContrastText	{59DDE9F2-5253-40EA-9A8B-479E96C6249A}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.Contrast. Not intended to be parsed programmatically.
System.Photo.DateTaken	{14B81DA1-0135-4D31-96D9-	36867	FALSE	TRUE		DateTime	8		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	6CBFC9671A99}								
System.Photo.DigitalZoom	{F85BF840-A925-4BC2-B0C4-8E36B598679E}	100	FALSE	TRUE		Double	8		Calculated from PKEY_Photo_DigitalZoomNumerator and PKEY_Photo_DigitalZoomDenominator
System.Photo.Event	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	18248	TRUE	TRUE		String	512	TRUE	The event at which the photo was taken.
System.Photo.ExposureBias	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	37380	FALSE	TRUE		Double	8		Calculated from PKEY_Photo_ExposureBiasNumerator and PKEY_Photo_ExposureBiasDenominator
System.Photo.ExposureProgram	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	34850	FALSE	TRUE		UInt32	4		name="Unknown" value="0" text="Unknown" name="Manual" value="1" text="Manual" name="Normal" value="2" text="Normal" name="Aperture" value="3" text="Aperture Priority" name="Shutter" value="4" text="Shutter Priority" name="Creative" value="5" text="Creative Program (biased toward depth of field)" name="Action" value="6" text="Action Program (biased toward shutter speed)" name="Portrait" value="7" text="Portrait Mode" name="Landscape" value="8" text="Landscape Mode"
System.Photo.ExposureProgramText	{FEC690B7-5F30-4646-AE47-	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.ExposureProgram. Not intended to be parsed programmatically.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	4CAAFBA884A3}								
System.Photo.ExposureTime	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	33434	FALSE	TRUE		Double	8		Calculated from PKEY_Photo_ExposureTimeNumerator and PKEY_Photo_ExposureTimeDenominator
System.Photo.Flash	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	37385	FALSE	TRUE		Byte	1		name="NoFlash" value="0" text="No flash" name="Flash" value="1" text="Flash" name="FlashNoReturnLight" value="5" text="Flash, no strobe return" name="FlashReturnLight" value="7" text="Flash, strobe return" name="FlashCompulsory" value="9" text="Flash, compulsory" name="FlashCompulsoryNoReturnLight" value="13" text="Flash, compulsory, no strobe return" name="FlashCompulsoryReturnLight" value="15" text="Flash, compulsory, strobe return" name="NoFlashCompulsory" value="16" text="No flash, compulsory" name="NoFlashAuto" value="24" text="No flash, auto" name="FlashAuto" value="25" text="Flash, auto" name="FlashAutoNoReturnLight" value="29" text="Flash, auto, no strobe return" name="FlashAutoReturnLight" value="31" text="Flash, auto, strobe return" name="NoFlashFunction" value="32" text="No flash function" name="FlashRedEye" value="65" text="Flash, red-eye" name="FlashRedEyeNoReturnLight" value="69" text="Flash, red-eye, no strobe return"

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
									name="FlashRedEyeReturnLight" value="71" text="Flash, red-eye, strobe return" name="FlashCompulsoryRedEye" value="73" text="Flash, compulsory, red-eye" name="FlashCompulsoryRedEyeNoReturnLight" value="77" text="Flash, compulsory, red-eye, no strobe return"name="FlashCompulsoryRedEyeReturnLight" value="79" text="Flash, compulsory, red-eye, strobe return" name="FlashAutoRedEye" value="89" text="Flash, auto, red-eye" name="FlashAutoRedEyeNoReturnLight" value="93" text="Flash, auto, no strobe return, red-eye" name="FlashAutoRedEyeReturnLight" value="95" text="Flash, auto, strobe return, red-eye"
System.Photo.FlashFired	{2D152B40-CA39-40DB-B2CC-573725B2FEC5}	100	FALSE	TRUE		Boolean	2		
System.Photo.FlashText	{6B8B68F6-200B-47EA-8D25-D8050F57339F}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.Flash. Not intended to be parsed programmatically.
System.Photo.FNumber	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	33437	FALSE	TRUE		Double	8		Calculated from PKEY_Photo_FnumberNumerator and PKEY_Photo_FnumberDenominator
System.Photo.FocalLength	{14B81DA1-0135-	3738	FALSE	TRUE		Doubl	8		Calculated from PKEY_Photo_FocalLengthNumerator and

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	4D31-96D9-6CBFC9671A99}	6				e			PKEY_Photo_FocalLengthDenominator
System.Photo.FocalLengthInFilm	{A0E74609-B84D-4F49-B860-462BD9971F98}	100	FALSE	TRUE		UInt16	2		
System.Photo.GainControlText	{C06238B2-0BF9-4279-A723-25856715CB9D}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.GainControl. Not intended to be parsed programmatically.
System.Photo.ISOSpeed	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	34855	FALSE	TRUE		UInt16	2		
System.Photo.LightSource	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	37384	FALSE	TRUE		UInt32	4		name="Unknown" value="0" text="Unknown" name="Daylight" value="1" text="Daylight" name="Fluorescent" value="2" text="Fluorescent" name="Tungsten" value="3" text="Tungsten" name="StandardA" value="17" text="Standard Illuminant A"name="StandardB" value="18" text="Standard Illuminant B"name="StandardC" value="19" text="Standard Illuminant C"name="D55" value="20" text="D55" name="D65" value="21" text="D65" name="D75" value="22" text="D75"
System.Photo.MaxAperture	{08F6D7C2-E3F2-	100	FALSE	TRUE		Double	8		Calculated from PKEY_Photo_MaxApertureNumerator and

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	44FC-AF1E-5AA5C81A2D3E}								PKEY_Photo_MaxApertureDenominator
System.Photo.MeteringMode	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	37383	FALSE	TRUE		UInt16	2		name="Unknown" value="0" text="Unknown" name="Average" value="1" text="Average" name="Center" value="2" text="Center Weighted Average" name="Spot" value="3" text="Spot" name="MultiSpot" value="4" text="Multi Spot" name="Pattern" value="5" text="Pattern" name="Partial" value="6" text="Partial"
System.Photo.MeteringModeText	{F628FD8C-7BA8-465A-A65B-C5AA79263A9E}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.MeteringMode. Not intended to be parsed programmatically.
System.Photo.Orientation	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	274	FALSE	TRUE		UInt16	2		This is the image orientation viewed in terms of rows and columns.name="Normal" value="1" text="Normal" name="FlipHorizontal" value="2" text="Flip horizontal" name="Rotate180" value="3" text="Rotate 180 degrees"name="FlipVertical" value="4" text="Flip vertical" name="Transpose" value="5" text="Transpose" name="Rotate270" value="6" text="Rotate 270 degrees"name="Transverse" value="7" text="Transverse" name="Rotate90" value="8" text="Rotate 90 degrees"
System.Photo.OrientationText	{A9EA193C-C511-498A-A06B-58E2776DC	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.Orientation. Not intended to be parsed programmatically.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	C28}								
System.Photo.PeopleNames	{E8309B6E-084C-49B4-B1FC-90A80331B638}	100	TRUE	TRUE		String	512	TRUE	The people tags on an image.
System.Photo.PhotometricInterpretationText	{821437D6-9EAB-4765-A589-3B1CBBD22A61}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.PhotometricInterpretation. Not intended to be parsed programmatically. Name="RGB" value="2" text="RGB" name="YcbCr" value="6" text="YcbCr"
System.Photo.ProgramModeText	{7FE3AA27-2648-42F3-89B0-454E5CB150C3}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.ProgramMode. Not intended to be parsed programmatically. Name="NotDefined" value="0" text="Not defined" name="Manual" value="1" text="Manual" name="Normal" value="2" text="Normal program" name="Aperture" value="3" text="Aperture priorityname="Shutter" value="4" text="Shutter priority"name="Creative" value="5" text="Creative program"name="Action" value="6" text="Action program" name="Portrait" value="7" text="Portrait" name="Landscape" value="8" text="Landscape"
System.Photo.SaturationText	{61478C08-B600-4A84-BBE4-E99C45F0A072}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.Saturation. Not intended to be parsed programmatically. Name="Normal" value="0" text="Normal" name="Low" value="1" text="Low saturation" name="High" value="2" text="High saturation"
System.Photo.SharpnessText	{51EC3F47-DD50-	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.Sharpness. Not intended to be parsed

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	421D-8769-334F50424B1E}								programmatically. Name="Normal" value="0" text="Normal"name="Soft" value="1" text="Soft" name="Hard" value="2" text="Hard"
System.Photo.ShutterSpeed	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	37377	FALSE	TRUE		Double	8		Calculated from PKEY_Photo_ShutterSpeedNumerator and PKEY_Photo_ShutterSpeedDenominator
System.Photo.SubjectDistance	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	37382	FALSE	TRUE		Double	8		Calculated from PKEY_Photo_SubjectDistanceNumerator and PKEY_Photo_SubjectDistanceDenominator
System.Photo.TagViewAggregate	{B812F15D-C2D8-4BBF-BACD-797443461113F}	100	TRUE	TRUE	I	String	512	TRUE	A read-only aggregation of tag-like properties for use in building views.
System.Photo.WhiteBalance	{EE3D3D8A-5381-4CFA-B13B-AAF66B5F4EC9}	100	FALSE	TRUE		UInt32	4		Indicates the white balance mode set when the image was shot. Name="Auto" value="0" text="Auto" name="Manual" value="1" text="Manual"
System.Photo.WhiteBalanceText	{6336B95E-C7A7-426D-86FD-7AE3D39C84B4}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Photo.WhiteBalance. Not intended to be parsed programmatically. Name="Low" value="0" text="Low" name="Normal" value="1" text="Normal"name="High" value="2" text="High"
System.Priority	{9C1FCF74-2D97-41BA-B4AE-	5	FALSE	TRUE		UInt16	2		name="Low" value="0" text="Low" name="Normal" value="1" text="Normal"name="High" value="2" text="High"

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	CB2E3661A6E4}								
System.PriorityText	{D98BE98B-B86B-4095-BF52-9D23B2E0A752}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Priority. Not intended to be parsed programmatically.
System.Project	{39A7F922-477C-48DE-8BC8-B28441E342E3}	100	TRUE	TRUE		String	512		
System.ProviderItemID	{F21D9941-81F0-471A-ADEE-4E74B49217ED}	100	TRUE	TRUE		String	512		
System.Rating	{64440492-4C8B-11D1-8B70-080036B11A03}	9	FALSE	TRUE		UInt32	4		Indicates the users preference rating of an item on a scale of 1-99 (1-12 = One Star, 13-37 = Two Stars, 38-62 = Three Stars, 63-87 = Four Stars, 88-99 = Five Stars).
System.RatingText	{90197CA7-FD8F-4E8C-9DA3-B57E1E609295}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Rating. Not intended to be parsed programmatically.
System.RecordedTV.ChannelNumber	{6D748DE2-8D38-4CC3-AC60-F009B	7	FALSE	TRUE		UInt32	4		Example: 42

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	057C557}								
System.RecordedTV.DateContentExpires	{6D748DE2-8D38-4CC3-AC60-F009B057C557}	15	FALSE	TRUE		DateTime	8		
System.RecordedTV.EpisodeName	{6D748DE2-8D38-4CC3-AC60-F009B057C557}	2	TRUE	TRUE		String	512		Example: "Nowhere to Hyde"
System.RecordedTV.IsATSCContent	{6D748DE2-8D38-4CC3-AC60-F009B057C557}	16	FALSE	TRUE		Boolean	2		
System.RecordedTV.IsClosedCaptioningAvailable	{6D748DE2-8D38-4CC3-AC60-F009B057C557}	12	FALSE	TRUE		Boolean	2		
System.RecordedTV.IsDTVContent	{6D748DE2-8D38-4CC3-AC60-F009B057C557}	17	FALSE	TRUE		Boolean	2		
System.RecordedTV.IsHDContent	{6D748DE2-8D38-	18	FALSE	TRUE		Boolean	2		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	4CC3-AC60-F009B057C557}								
System.RecordedTV.IsRepeatBroadcast	{6D748DE2-8D38-4CC3-AC60-F009B057C557}	13	FALSE	TRUE		Boolean	2		
System.RecordedTV.IsSAP	{6D748DE2-8D38-4CC3-AC60-F009B057C557}	14	FALSE	TRUE		Boolean	2		
System.RecordedTV.NetworkAffiliation	{2C53C813-FB63-4E22-A1AB-0B331CA1E273}	100	FALSE	TRUE		String	512		
System.RecordedTV.OriginalBroadcastDate	{4684FE97-8765-4842-9C13-F006447B178C}	100	FALSE	TRUE		DateTime	8		
System.RecordedTV.ProgramDescription	{6D748DE2-8D38-4CC3-AC60-F009B057C557}	3	TRUE	TRUE		String	2048		
System.RecordedTV.RecordingTime	{A5477F61-7A82-	100	FALSE	TRUE		DateTime	8		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	4ECA-9DDE-98B69B2479B3}								
System.RecordedTV.StationCallSign	{6D748DE2-8D38-4CC3-AC60-F009B057C557}	5	TRUE	TRUE		String	512		Example: "TOONP"
System.RecordedTV.StationName	{1B5439E7-EBA1-4AF8-BDD7-7AF1D4549493}	100	TRUE	TRUE		String	512		
System.SDID	{B725F130-47EF-101A-A5F1-02608C9EEBAC}	LEGACY	FALSE	TRUE		UInt32	4		Internal MSSearch Security Descriptor ID
System.Search.AccessCount	{0B63E350-9CCC-11D0-BCDB-00805FCCCE04}	9	FALSE	TRUE		UInt32	4		The number of times that the Windows Search Gatherer process pushed properties of this document to the Windows Search Gatherer Plugins.
System.Search.AutoSummary	{560C36C0-503A-11CF-BAA1-00004C752A9A}	2	FALSE	TRUE	NotIndexed	String	2048		General Summary of the document.
System.Search.ContainerHash	{BCEEE283-35DF-4D53-	100	TRUE	FALSE		String	512		Hash code used to identify attachments to be deleted based on a common container URL.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	826A-F36A3EEFC6BE}								
System.Search.Contents	{B725F130-47EF-101A-A5F1-02608C9EEBAC}	19	TRUE	FALSE		String	512		The contents of the item. This property is for query restrictions only; it cannot be retrieved in a query result. The WSS friendly name is 'contents'.
System.Search.EntryID	{49691C90-7E17-101A-A91C-08002B2ECDA9}	5	FALSE	TRUE	NotIndexed	Int32	4		The entry ID for an item within a given catalog in the Windows Search Index. This value can be recycled, and therefore is not considered unique over time.
System.Search.GatherTime	{0B63E350-9CCC-11D0-BCDB-00805FCCCE04}	8	FALSE	TRUE		DateTime	8		The Datetime that the Windows Search Gatherer process last pushed properties of this document to the Windows Search Gatherer Plugins.
System.Search.HitCount	{49691C90-7E17-101A-A91C-08002B2ECDA9}	4	FALSE	TRUE	NotIndexed	Int32	4		When using CONTAINS over the Windows Search Index, this is the number of matches of the term. If there are multiple CONTAINS, an AND computes the min number of hits and an OR the max number of hits.
System.Search.LastIndexedTotalTime	{0B63E350-9CCC-11D0-BCDB-00805FCCCE04}	11	FALSE	TRUE		Double	8		The total time in seconds taken to index this document the last time it was indexed.
System.Search.Rank	{49691C90-7E17-101A-A91C-08002B2EC	3	FALSE	TRUE	NotIndexed	Int32	4		Relevance rank of row. Ranges from 0-1000. Larger numbers = better matches. Query-time only.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	DA9}								
System.Search.ReverseFileName	{49691C90-7E17-101A-A91C-08002B2ECDA9}	8	TRUE	TRUE		String	520		Reverse of the FileName from Query propset.
System.Search.Store	{A06992B3-8CAF-4ED7-A547-B259E32AC9FC}	100	TRUE	TRUE		String	512		The identifier for the protocol handler that produced this item. (E.g. MAPI, CSC, FILE etc.)
System.Search.Scope	{B725F130-47EF-101A-A5F1-02608C9EEBAC }	22	FALSE	FALSE		String			Used to narrow the scope of a query to the specified directory and subdirectories.
System.ItemStoragePathDeprecated	{B725F130-47EF-101A-A5F1-02608C9EEBAC }	11	FALSE	FALSE		String			Deprecated.
System.Search.RowID	{49691C90-7E17-101A-A91C-08002B2ECDA9 }	15	FALSE	FALSE		Int32			The entry ID for a row in a grouped rowset.
System.Sensitivity	{F8D3F6AC-4874-42CB-BE59-AB454B3071	100	FALSE	TRUE		UInt16	2		name="Normal" value="0" text="Normal" name="Personal" value="1" text="Personal" name="Private" value="2" text="Private" name="Confidential" value="3"

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	6A}								text="Confidential"
System.SensitivityText	{D0C7F054-3F72-4725-8527-129A577CB269}	100	TRUE	TRUE		String	512		This is the user-friendly form of System.Sensitivity. Not intended to be parsed programmatically.
System.SFGAOFlags	{28636AA6-953D-11D2-B5D6-00C04FD918D0}	25	FALSE	TRUE		UInt32	4		IshellFolder::GetAttributesOf flags, with SFGAO_PKEYSFGAOMASK attributes masked out.
System.Shell.OmitFromView	{DE35258C-C695-4CBC-B982-38B0AD24CED0}	2	TRUE	TRUE		String	512		Set this to a string value of 'True' to omit this item from shell views
System.Shell.SFGAOFlagsStrings	{D6942081-D53B-443D-AD47-5E059D9CD27A}	2	TRUE	FALSE		String	512	TRUE	Expresses the SFGAO flags as string values and is used as a query optimization.name="FileSys" value="filesys" name="FileSysAncestor" value="fileanc" name="StorageAncestor" value="storageanc"name="Stream" value="stream" name="Link" value="link" name="Hidden" value="hidden" name="Superhidden" value="superhiddenname="Folder" value="folder" name="NonEnumerated" value="nonenum" name="Browsable" value="browsable"
System.Size	{B725F130-47EF-101A-A5F1-	12	FALSE	TRUE		UInt64	8		name="Empty" minValue="0" setValue="0" text="Empty (0 KB)" name="Tiny" minValue="1" setValue="1" text="Tiny (0

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	02608C9EEBAC}								– 10 KB)" dname="Small" minValue="10241" setValue="10241" text="Small (10 – 100 KB)" name="Medium" minValue="102401" setValue="102401" text="Medium (100 KB – 1 MB)"name="Large" minValue="1048577" setValue="1048577" text="Large (1 – 16 MB)" name="Huge" minValue="16777217" setValue="16777217" text="Huge (16 – 128 MB)" name="Gigantic" minValue="134217729" setValue="134217729" text="Gigantic (>128 MB)"
System.Software.DateLastUsed	{841E4F90-FF59-4D16-8947-E81BBFFAB36D}	16	TRUE	TRUE		DateTime	8		
System.Software.ProductName	{0CEF7D53-FA64-11D1-A203-0000F81FEDEE}	7	TRUE	FALSE		String	512		
System.Software.ProductVersion	{0CEF7D53-FA64-11D1-A203-0000F81FEDEE}	8	TRUE	FALSE		String	512		
System.SoftwareUsed	{14B81DA1-0135-4D31-96D9-6CBFC9671A99}	305	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.SourceItem	{668CDFA5-7A1B-4323-AE4B-E527393A1D81}	100	TRUE	TRUE		String	512		
System.StartDate	{48FD6EC8-8A12-4CDF-A03E-4EC5A511EDDE}	100	FALSE	TRUE		DateTime	8		
System.Status	{00021A1-0000-0000-C000-000000000046}	9	TRUE	TRUE		String	512		
System.Subject	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	3	TRUE	TRUE		String	520		
System.Task.BillingInformation	{D37D52C6-261C-4303-82B3-08B926AC6F12}	100	TRUE	FALSE		String	512		
System.Task.CompletionStatus	{084D8A0A-E6D5-40DE-BF1F-C8820E7C877C}	100	TRUE	TRUE		String	512		

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Task.Owner	{08C7CC5F-60F2-4494-AD75-55E3E0B5ADD0}	100	TRUE	TRUE		String	256		
System.ThumbnailCacheId	{446D16B1-8DAD-4870-A748-402EA43D788C}	100	FALSE	TRUE		UInt64	8		Unique value that can be used as a key to cache thumbnails. The value changes when the name, volume, or data modified of an item changes.
System.Title	{F29F85E0-4FF9-1068-AB91-08002B27B3D9}	2	TRUE	TRUE		String	520		
System.Video.Compression	{64440491-4C8B-11D1-8B70-080036B11A03}	10	TRUE	TRUE		String	512		Indicates the level of compression for the video stream. "Compression".
System.Video.Director	{64440492-4C8B-11D1-8B70-080036B11A03}	20	TRUE	TRUE		String	256	TRUE	
System.Video.EncodingBitrate	{64440491-4C8B-11D1-8B70-080036B11A03}	8	FALSE	TRUE		UInt32	4		Indicates the data rate in "bits per second" for the video stream. "DataRate".

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
System.Video.FourCC	{64440491-4C8B-11D1-8B70-080036B11A03}	44	FALSE	TRUE		UInt32	4		Indicates the 4CC for the video stream.
System.Video.FrameHeight	{64440491-4C8B-11D1-8B70-080036B11A03}	4	FALSE	TRUE		UInt32	4		Indicates the frame height for the video stream.
System.Video.FrameRate	{64440491-4C8B-11D1-8B70-080036B11A03}	6	FALSE	TRUE		UInt32	4		Indicates the frame rate in "frames per millisecond" for the video stream.
System.Video.FrameWidth	{64440491-4C8B-11D1-8B70-080036B11A03}	3	FALSE	TRUE		UInt32	4		Indicates the frame width for the video stream.
System.Video.HorizontalAspectRatio	{64440491-4C8B-11D1-8B70-080036B11A03}	42	FALSE	TRUE		UInt32	4		Indicates the horizontal portion of the aspect ratio. The X portion of XX:YY, like 16:9.
System.Video.SampleSize	{64440491-4C8B-11D1-8B70-080036B11A03}	9	FALSE	TRUE		UInt32	4		Indicates the sample size in bits for the video stream. "SampleSize".
System.Video.StreamName	{64440491-4C8B-	2	TRUE	TRUE		String	512		Indicates the name for the video stream.

Property Name	GUID	propID	inInvertedIndex	isColumn	columnIndexType	type	MaxSize	VectorProperty	Description
	11D1-8B70-080036B11A03}								
System.Video.TotalBitrate	{64440491-4C8B-11D1-8B70-080036B11A03}	43	FALSE	TRUE		UInt32	4		Indicates the total data rate in "bits per second" for all video and audio streams.
System.Video.VerticalAspectRatio	{64440491-4C8B-11D1-8B70-080036B11A03}	45	FALSE	TRUE		UInt32	4		Indicates the vertical portion of the aspect ratio. The Y portion of XX:YY.

3 Protocol Details

The Windows Search Protocol message requests require only minimal sequencing. All messages MUST be preceded by an initial CPMConnectIn message (for example, at least one CPMConnectIn for each named pipe connection). Beyond the initial connection there is no other over all sequencing required by the protocol. Some messages do, however, have prerequisite messages that must be sent first. For more information, see the table in section 3.1.5. It is advised that the higher layer adhere to a meaningful message sequence, because the server will respond with an error for messages that are received without the prerequisite message or with invalid data. Note that some messages are also dependent on the higher-layer providing valid data that was received in messages earlier in a sequence.<24>

A typical message sequence for a simple query from a client to a remote computer is illustrated in the following diagram.

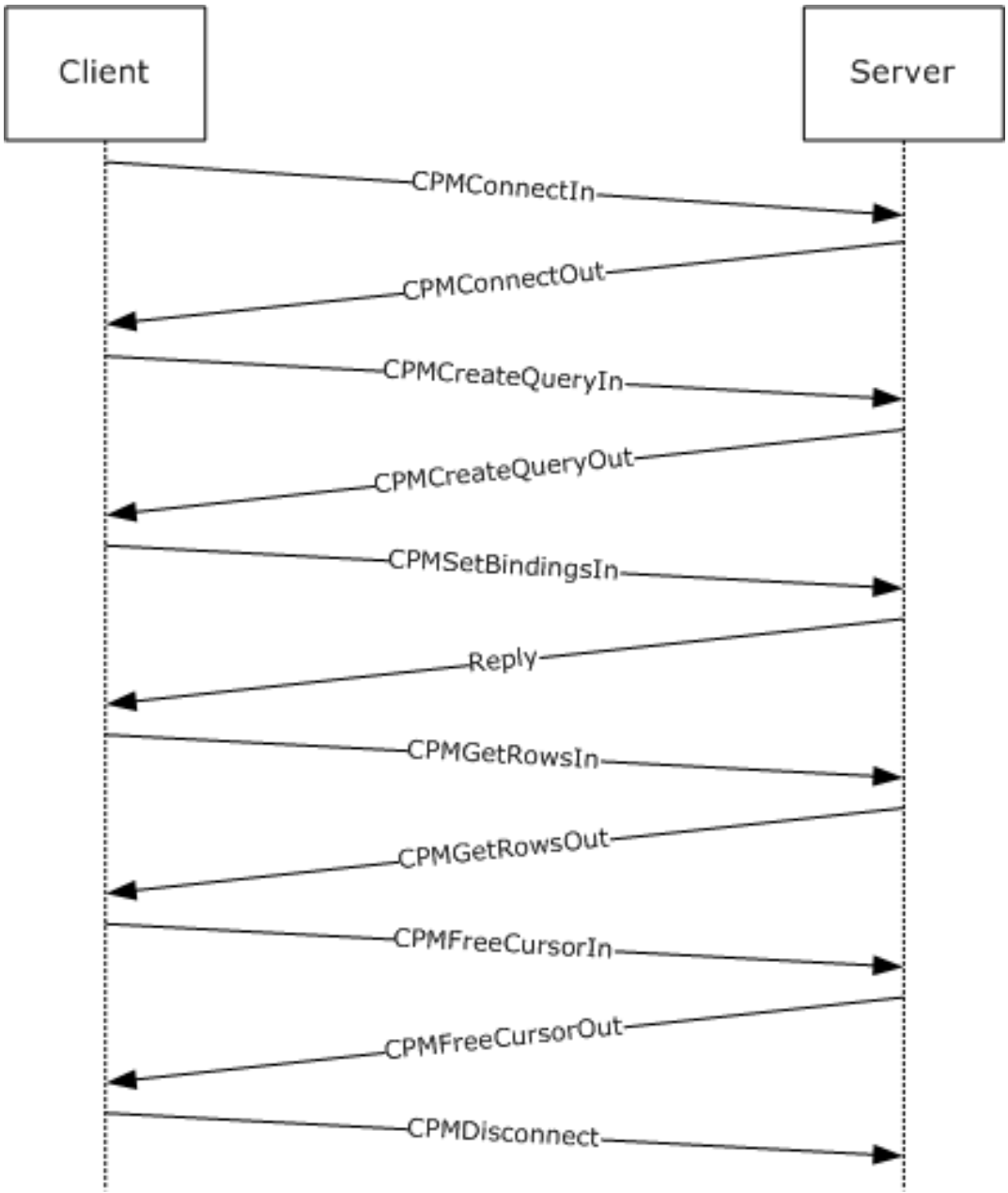


Figure 4: Windows Search Protocol session life cycle

The messages represented in the preceding diagram represent a subset of all of the Windows Search Protocol messages used for querying a remote GSS catalog.

3.1 Server Details

3.1.1 Abstract Data Model

The following section specifies data and state maintained by the Windows Search Protocol server. The data provided in this document explains how the protocol behaves. This section does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The Generic Search Service (GSS) implementing the Windows Search Protocol MUST maintain the following abstract data elements:

ConnectedClientsIdentifiers: A list of 32-bit unsigned integers. Values are added upon successful responses to CPMConnectIn messages and are removed when the clients disconnect. These integer identifiers are actually set to the values of the named pipe HANDLEs to connected clients because they uniquely identify incoming connections. The identifiers are removed from the list when the named pipe with the same HANDLEs are disconnected. Apart from the unique connection identification quality, no named pipe HANDLE semantics are assumed about the integer values in this list. Queries can be executed only one at a time over any given connection. Therefore, the server passes values in the **ConnectedClientIdentifiers** list as the *QueryIdentifier* argument to any GSS abstract interface calls that it makes.

ConnectedClientVersions: A list of 32-bit unsigned integers, one for each identifier in the **ConnectedClientIdentifiers** list. If the last 2 bytes of the client version are greater than or equal to 0x109, the server will verify the checksums in the message. If the version is greater than 0x10000, the server detects that the client is a 64-bit system.

3.1.2 Timers

The following timer is required for the server.

EventTimer: This is a periodic timer that is started upon receipt of a CPMSetScopePrioritizationIn message. The timer expiration interval is defined by the server when the timer is started. This timer instance is associated with exactly one active query. There can be multiple EventTimers, one for each query.

The following input or state is required for this timer.

QueryIdentifier: A unique query identifier of the query for which this timer was started.

3.1.3 Initialization

Upon initialization, the server MUST set its state to "not initialized" and start listening for messages on the named pipe specified in section 1.9. After performing any other internal initialization, it MUST transition to the "running" state.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Processing and Sequencing Rules

Whenever an error occurs during processing of a message sent by a client, the server MUST report an error back to the client as follows:

- Stop processing the message sent by the client.
- Respond with the message header (only) of the message sent by the client, keeping the **_msg** field intact. The only exception to this is when an error occurs in the CPMConnectIn (section 2.2.3.2) request and the server cannot find a catalog. The response is a CPMConnectOut (section 2.2.3.3) message with the following possible errors.

Error	Meaning
DS_E_DATASOURCENOTAVAILABLE	Catalog not in ready state for queries.

Error	Meaning
DS_E_INVALIDDATASOURCE	Failed to specify catalog correctly.
CI_E_NO_CATALOG	Catalog not found.

- Set the **_status** field to the error code value.

When a message arrives, the server MUST check the **_msg** field value to identify whether it is a known type (see section 2.2.2). If the type is not known, it MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

The server MUST then validate the **_ulChecksum** field value if the message type is one of the following:

- CPMConnectIn (0x000000C8)
- CPMCreateQueryIn (0x000000CA)
- CPMSetBindingsIn (0x000000D0)
- CPMGetRowsIn (0x000000CC)
- CPMFetchValueIn (0x000000E4)

To validate the **_ulChecksum** field value, the server MUST check the value that the client specified in the **_iClientVersion** field in the CPMConnectIn message.

If the value of the **_iClientVersion** field's last 2 bytes is 0x00000109 or greater and **_ulChecksum** is not equal to zero, the server MUST validate that the **_ulChecksum** field was calculated as specified in section 3.2.4. If the **_ulChecksum** value is invalid, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

Next, the server checks which state it is in. If its state is "not initialized", the server MUST report a CI_E_NOT_INITIALIZED (0x8004180B) error. If the state is "shutting down", the server MUST report a CI_E_SHUTDOWN (0x80041812) error.

After a header has been determined to be valid and the server to be in "running" state, further message-specific processing MUST be performed, as specified in the following subsections.

Some messages are valid only after a previous message has been sent. Typically, an ID or handle from the earlier message is required as input to the later message. These requirements are detailed in the following sections. The table below summarizes the relationship between messages.

Message	Prerequisite Message																
	CPMCiStateInOut	CPMUpdateDocumentsIn	CPMForceMergeIn	CPMConnectOut	CPMCreateQueryOut	CPMGetQueryStatusOut	CPMGetQueryStatusExOut	CPMSetBindingsOut	CPMGetRowsOut	CPMRatioFinishedOut	CPMFetchValueOut	CPMGetNotify	CPMSendNotifyOut	CPMGetApproximatePositionOut	CPMCompareBmkOut	CPMRestartPositionOut	CPMFreeCursorOut
CPMCiStateInOut				X													
CPMUpdateDocumentsIn				X													
CPMForceMergeIn				X													
CPMConnectIn																	
CPMCreateQueryIn				X													
CPMGetQueryStatusIn				X	X												
CPMGetQueryStatusExIn				X	X												
CPMSetBindingsIn				X	X												
CPMGetRowsIn				X	X			X									
CPMRatioFinishedIn				X	X												
CPMFetchValueIn				X	X			X	X								
CPMGetNotify				X	X												
CPMSendNotifyOut				X	X							X					
CPMGetApproximatePositionIn				X	X												
CPMCompareBmkIn				X	X												
CPMRestartPositionIn				X	X												
CPMStopAsynch				X	X												
CPMFreeCursorIn				X	X												
CPMDisconnect																	
CPMFindIndicesIn				X	X												
CPMGetRowsetNotifyIn				X	X												
CPMSetScopePrioritizationIn				X	X												
CPMGetScopeStatisticsIn				X	X												

Figure 5: Windows Search Protocol message sequence relationships

3.1.5.1 Remote Windows Search Service Catalog Management

3.1.5.1.1 Receiving a CPMCiStateInOut Request

When the server receives a CPMCiStateInOut message request from the client, the server MUST first search the **ConnectedClientIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMCiStateInOut message. If the handle is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error; otherwise, it MUST respond to the client with a

CPMCiStateInOut message containing information about the client's associated catalog, obtained by calling the **GetState** abstract interface.

3.1.5.2 Remote Windows Search Service Querying

3.1.5.2.1 Receiving a CPMConnectIn Request

When the server receives a CPMConnectIn request from a client, the server MUST do the following.

1. Search the **ConnectedClientIdentifiers** list for the handle of the named pipe over which the server has received the CPMConnectIn message. If it is present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Call the **IsCatalogAvailable** abstract interface. Provide the DBPROP_CI_CATALOG_NAME property value from the CPMConnectIn request message as the sole argument. Report a MSS_E_CATALOGNOTFOUND (0x80042103) error if the result of the **IsCatalogAvailable** abstract interface call to the GSS is not true. <25>

3. If the value of the **iClientVersion** field is larger than 0x00000109, then validate the checksum of the CPMConnectIn request as described in section 3.2.4.

4. Add the handle of the named pipe over which the server received the CPMConnectIn message to the **ConnectedClientIdentifiers** list. Call the StoreClientInformation abstract interface to the GSS with the query identifier that uniquely identifies the query to the server, the CPMConnectIn message as the *ConnectMessage* argument, and the HANDLE of the named pipe over which the server has received the CPMConnectIn message as the *NamedPipeHandle* argument. This helps the GSS later assess whether the user identified here has access to returned results.

5. Store the _**iClientVersion** field in the **ConnectedClientVersions** list under the same index as the index in the **ConnectedClientIdentifiers** list of the handle of the named pipe over which the server received this message.

6. Call the **GetServerVersions** abstract interface of the GSS. Fill in the **_serverVersion** field using the **_serverVersion** output argument obtained from the **GetServerVersions** abstract interface. If **supportsVersioningInfo** is true, fill in the **dwWinVerMajor**, **dwWinVerMinor**, **dwNLSVerMajor**, and **dwNLSVerMinor** fields with the values returned by **GetServerVersions**. Otherwise, in order to indicate that versioning is not supported (see section 3.2.4.2.1), the server MUST copy four DWORDs at the offset starting after **_serverVersion** in the CPMConnectIn message to the same location in the CPMConnectOut reply (the offset right after **_serverVersion**).

7. Respond to the client with the CPMConnectOut message.

8. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: Generated by any resource allocation failure on the server or service side.

   - STATUS_INVALID_PARAMETER: Generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   - STATUS_INVALID_PARAMETER_MIX: Generated when the client version as passed in this message is smaller than CI_VERSION_WDS30 (0x102).

   - DS_E_DATASOURCENOTAVAILABLE: Generated when the catalog requested is not in a ready state for queries.

   - DS_E_ INVALIDDATASOURCE: Generated when the catalog was not specified correctly.

- CI_E_NO_CATALOG: Generated when the catalog requested was not found.

  Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.2 Receiving a CPMCreateQueryIn Request

When the server receives a CPMCreateQueryIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMCreateQueryIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Run the **RunNewQuery** abstract interface of the GSS. Pass in as parameters the HANDLE of the named pipe over which the server has received the CPMCreateQueryIn message, along with the **ColumnSet**, **RestrictionArray**, **SortSet**, **CCategorizationSet**, **RowSetProperties**, **PidMapper**, **GroupArray**, and **Lcid** values. If the *CanRunQueryNow* output parameter is not true, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error. Otherwise, report the *QueryParametersError* output parameter in the reply. Details for how to report an error are specified in section 3.1.5.

3. Respond to the client with a CPMCreateQueryOut message by using the *CursorHandlesList*, *fTrueSequential*, and *fWorkidUnique* outputs obtained from the **RunNewQuery** abstract interface to the GSS.

4. Report any errors encountered during message preparation or during any abstract interface call to the GSS. The following errors are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.

   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client are invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   - STATUS_INVALID_PARAMETER_MIX: generated when the client version, as passed in this message, is smaller than CI_VERSION_WDS30 (0x102).

   - STATUS_NO_MEMORY: generated on memory allocation errors.

   - STATUS_INSUFFICIENT_RESOURCES: generated on resource allocation errors. These resources could be file or pipe handles, any data structures that are specific to the implementation of the GSS, or any other resources used by the GSS implementation.

   - CI_E_NOT_FOUND: generated when the requested property was not found.

   - ERROR_INVALID_PARAMETER: this is generated by the WSS implementation on some internal errors, such as registry access failures. Implementations of the GSS can choose to do the same.

   - CI_E_TIMEOUT: generated on named pipe communication timeout.

   - E_ACCESSDENIED: generated when the client does not have permissions to access a needed resource, such as a file result or a catalog.

     Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.3 Receiving a CPMGetQueryStatusIn Request

When the server receives a CPMGetQueryStatusIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMGetQueryStatusIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Call the **ClientQueryHasCursorHandle** abstract interface to the GSS with the HANDLE of the named pipe over which the server has received the CPMGetQueryStatusIn message as its *QueryIdentifier* argument and with the *_hCursor* handle as its *CursorHandle* argument. <26>

3. Call the **GetQueryStatus** abstract interface to the GSS. Use the HANDLE of the named pipe over which the server has received the CPMGetQueryStatusIn message as its *QueryIdentifier* argument. Prepare a CPMGetQueryStatusOut message, using the *QueryStatus* output for the **_QStatus** field. If the Error output is not 0, then report the error.

4. Respond to the client with the CPMGetQueryStatusOut message.

5. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.4 Receiving a CPMGetQueryStatusExIn Request

When the server receives a CPMGetQueryStatusExIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMGetQueryStatusExIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Call the **ClientQueryHasCursorHandle** abstract interface to the GSS with the HANDLE of the named pipe over which the server has received the CPMGetQueryStatusIn message as its *QueryIdentifier* argument and with the *_hCursor* handle as its *CursorHandle* argument.<27>

3. Prepare a CPMGetQueryStatusExOut message, using corresponding parameters obtained by calling the following abstract interfaces to the GSS.

   - **GetState**, with no parameters
     - Parameters: none
     - Use Output parameters:
       - cFilteredDocuments = CFilteredDocuments
       - Set cDocumentsToFilter to: CTotalDocuments - CFilteredDocuments
   - **GetQueryStatus**
     - Parameters:
       - The HANDLE of the named pipe over which the server has received the CPMGetQueryStatusExIn message.
     - Use Output parameters:

        - QStatus = QStatus
    - **GetRatioFinishedParams**
        - Parameters:
            - The HANDLE of the named pipe over which the server has received the CPMGetQueryStatusExIn message.
            - hCursor
        - Use Output parameters:
            - dwRatioFinishedDenominator = rdwRatioFinishedDenominator
            - dwRatioFinishedNumerator = rdwRatioFinishedNumerator
    - **GetApproximatePosition**
        - Parameters:
            - The HANDLE of the named pipe over which the server has received the CPMGetQueryStatusExIn message.
            - hCursor
            - bmk
        - Use Output Parameters:
            - iRowBmk = riRowBmk
    - **GetWhereID**
        - Parameters:
            - The HANDLE of the named pipe over which the server has received the CPMGetQueryStatusExIn message.
        - Use Output parameters:
            - whereID = whereid
    - **GetExpensiveProperties**
        - Parameters:
            - The HANDLE of the named pipe over which the server has received the CPMGetQueryStatusExIn message.
            - hCursor
        - Use Output parameters:
            - cRowsTotal = rcRowsTotal
            - cResultsFound = rdwResultCount
            - maxRank = maxRank
4. Respond to the client with the CPMGetQueryStatusExOut message.

5. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.
   - E_ACCESSDENIED: generated when the client does not have permissions to access a needed resource such as a file result or a catalog.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.5 Receiving a CPMRatioFinishedIn Request

When the server receives a CPMRatioFinishedIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMRatioFinishedIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.
2. Call the **ClientQueryHasCursorHandle** abstract interface to the GSS with the HANDLE of the named pipe over which the server has received the CPMRatioFinishedIn message as its *QueryIdentifier* argument and with the *hCursor* handle as its *CursorHandle* argument. <28>
3. Call the **GetRatioFinishedParams** abstract interface with the HANDLE of the named pipe over which the server has received the CPMRatioFinishedIn message as its *QueryIdentifier* argument and with the *hCursor* handle as its *CursorHandle* argument.
4. Prepare a CPMRatioFinishedOut message; the server MUST set the CPMRatioFinishedOut parameters as follows.

   - ulNumerator = rdwRatioFinishedNumerator
   - ulDenominator = rdwRatioFinishedDenominator
   - cRows = cRows
   - fNewRows = fNewRows

   If this step fails for any reason, the server MUST report any error code encountered in performing the request in accordance with Win32 Error Codes in [MS-ERREF].
5. Respond to the client with the CPMRatioFinishedOut message.
6. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.
   - E_ACCESSDENIED: generated when the client does not have permissions to access a needed resource such as a file result or a catalog.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.6 Receiving a CPMGetRowsIn Request

When the server receives a CPMGetRowsIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMGetRowsIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Call the **ClientQueryHasCursorHandle** abstract interface to the GSS with the HANDLE of the named pipe over which the server has received the CPMGetRowsIn message as its *QueryIdentifier* argument and with the *_hCursor* handle as its *CursorHandle* argument.<29>

3. Call the ***HasBindings*** abstract interface with the HANDLE of the named pipe over which the server has received the CPMGetRowsIn message as its *QueryIdentifier* argument and with the *_hCursor* handle as its *CursorHandle* argument. If the *hasBindings* output parameter is not true, the server MUST report an E_UNEXPECTED (0x8000FFFF) error.<30>

4. Prepare a CPMGetRowsOut message by setting the position from which to retrieve the next rows, according to the received **SeekDescription**:

    - If eType == 0x00000000, then no work needs to be done to reposition the next row's read index within the rowset.

    - Define oldIndex as the result of calling the **GetNextGetRowsPosition** abstract interface with *QueryIdentifier* and *CursorHandle* as arguments. Then, as a function of eType, call the **SetNextGetRowsPosition** abstract interface with *QueryIdentifier*, *chapter*, and *CursorHandle* as arguments and with Index set to:

        - If eType == eRowSeekNext (section 2.2.3.11), then Index = oldIndex + cskip (section 2.2.1.37).

        - If eType == eRowSeekAt (section 2.2.3.11), then Index = GetBookmarkPosition(QueryIdentifier, CursorHandle, _bmkOffset) + _cskip (sections 3.1.7and 2.2.1.37).

        - If eType == eRowSeekatRatio (section 2.2.3.11), then Index = (ulNumerator/ulDenominator) * rcRowsTotal. The value of *rcRowsTotal* (section 3.1.7) is equal to the output argument of the **GetExpensiveProperties** abstract interface called with *QueryIdentifier* and *CursorHandle* as arguments.

            - When the value of ulDenominator is zero, or if _ulDenominator > _ulNumerator, DB_E_BADRATIO is returned as the error value.

    - If eType == eRowSeekByBookmark (section 2.2.3.11), then for each bookmark handle value in the **aBookmarks** array (section 2.2.1.39) of the SeekDescription (section 2.2.3.11) argument:

        - Call the **GetBookmarkPosition** abstract interface to the GSS with *QueryIdentifier*, *CursorHandle*, and the bookmark handle value as arguments.

        - Use the *bmkIndex* (section 3.1.7) returned as an argument to **SetNextGetRowsPosition** (section 3.1.7), along with *QueryIdentifier*, *CursorHandle*, and *_chapter*.

        - Call the **GetRows** abstract interface with *QueryIdentifier*, *CursorHandle*, 1, and *_fBwdFetch* as arguments.

5. After the position is set, retrieve the desired row from the GSS by calling the **GetRows** abstract interface with the HANDLE of the named pipe over which the server has received the CPMGetRowsIn message as its *QueryIdentifier* argument, with the *_hCursor* handle as its

CursorHandle argument, with *_chapter* as its *chapter* argument, with *_cRowsToTransfer* as its *NumRowsRequested* argument, and with *_fBwdFetch* as its *FetchForward* argument. Do this in all cases, except for step 4 bullet 3.

6. Copy as many rows as fit in a buffer, the size of which is indicated by **_cbReadBuffer** (section 2.2.3.11), but not more than indicated by **_cRowsToTransfer** (section 2.2.3.11). Thereafter, reposition the cursor to reflect the actual number of returned rows by calling the **SetNextGetRowsPosition** abstract interface with *QueryIdentifier*, *CursorHandle* and *_chapter* as arguments and with Index set to the old index (as obtained by calling **GetNextRowsPosition**(QueryIdentifier, _hCursor, _chapter)) plus the number of rows that fit in the buffer.

7. If there are no more rows available on the server to finish this request, as signaled by the *NoMoreRowsToReturn* output parameter for the call made to the **GetRows** abstract interface (section 3.1.7), the **Status** fleld of CPMGetRowsOut message MUST be set to DB_S_ENDOFROWSET.

8. Store the number of rows fetched in **_cRowsReturned** (section 2.2.3.12), as determined by the *NumRowsReturned* output parameter from the **GetRows** call (or, in the case of step 4 bullet 3, the sum of the *NumRowsReturned* parameters from the **GetRows** calls).

9. Copy the **_chapt** field from the CPMGetRowsIn message to a CPMGetRowsOut message to be sent.

10. If either of the following cases are true, copy the **SeekDescription** (section 2.2.3.11) from the CPMGetRowsIn to the CPMGetRowsOut message.

    - The server fails to completely fill the buffer due to a memory allocation failure, but has been able to store at least one row.

      Copying the **SeekDescription** (section 2.2.3.11) allows the client to continue from the point where the server left off. In addition, the server SHOULD set the error code DB_S_BLOCKLIMITEDROWS, and in the case of CRowSeekAtRatio, convert CRowSeekAtRatio to CRowSeekAt.

    - CRowSeekByBookmark is specified.

    Otherwise, clear SeekDescription (section 2.2.3.11) by setting it to zero.

11. Store the fetched rows in the **Rows** field (see section 2.2.3.12 for details on the structure of the **Rows** field).

    Note Regarding status byte field: If StatusUsed is set to 0x01 in the CTableColumn of the CPMSetBindingsIn message for the column, the server MUST set the status byte (which is located at StatusOffset from the start of the rows) for this column to one of the following values.

Value	Meaning
0x00	StoreStatusOK
0x01	StoreStatusDeferred
0x02	StoreStatusNull

12. Respond to the client with the CPMGetRowsOut message.

    If the property value is absent for this row, the server MUST set the status byte to StoreStatusNull. If the value is too big to be transferred in the CPMGetRowsOut message (greater than 2048 bytes), the server MUST set the status byte to StoreStatusDeferred. Otherwise, the server MUST set the status byte to StoreStatusOK.

13. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

    - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
    - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.
    - STATUS_NO_MEMORY: generated on memory allocation errors.
    - CI_E_NOT_FOUND: generated when the requested property was not found.
    - E_ACCESSDENIED: generated when the client does not have permissions to access a needed resource such as a file result or a catalog.
    - CI_E_BUFFERTOOSMALL: generated when the buffer passed in is too small to accommodate the requested property or properties. This error signals the client to request the potentially large property value separately using a CPMFetchValueIn request.

    Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.7 Receiving a CPMFetchValueIn Request

When the server receives a CPMFetchValueIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMFetchValueIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.
2. Prepare a CPMFetchValueOut message. If this step fails for any reason, the server MUST report an error.
3. Call the **GetPropertyValueForWorkid** abstract interface with the HANDLE of the named pipe over which the server has received the CPMFetchValueIn message, *_wid*, and *Propspec* as arguments. If the *ValueExists* output parameter is not true, the server MUST set **_fValueExists** (section 2.2.3.16) to 0x00000000; otherwise set **_fValueExists** (section 2.2.3.16) to 0x00000001. If this step fails for any reason, the server MUST report any error code encountered in performing the request in accordance with Win32 Error Codes in [MS-ERREF].
4. If **_fValueExists** is equal to 0x00000001, the server MUST do the following:

    - Scan the *Property* output parameter structure, starting from the **_cbSoFar** offset, and copy a maximum of **_cbChunk** bytes (do not go past the end of the serialized property) to the **vValue** field. If this step fails for any reason, the server MUST report an error.
    - Set **_cbValue** to the number of bytes copied in the previous step.
    - If the length of the serialized property is greater than **_cbSoFar** added to **_cbValue**, set **_fMoreExists** to 0x00000001; otherwise, set it to 0x00000000.

5. Respond to the client with the CPMFetchValueOut message.
6. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

    - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.

- STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.
- E_ACCESSDENIED: generated when the client does not have permissions to access a needed resource such as a file result or a catalog.
- CI_E_BUFFERTOOSMALL: generated when the buffer passed in is too small to accommodate the requested property. This error signals the client to request the potentially large property value separately, using a CPMFetchValueIn request with a larger buffer.

Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.8 Receiving a CPMSetBindingsIn Request

When the server receives a CPMSetBindingsIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMSetBindingsIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.
2. Call the **ClientQueryHasCursorHandle** abstract interface to the GSS with the HANDLE of the named pipe over which the server has received the CPMSetBindingsIn message as its *QueryIdentifier* argument and with the **_hCursor** handle as its *CursorHandle* argument.<31>
3. Verify that the bindings information is valid (that is, the column at least specifies value, length, or status to be returned; there is no overlap in bindings for value, length, or status; and value, length, and status fit in the row size specified) and, if not, report a DB_E_BADBINDINFO (0x80040E08) error.<32>
4. Call the **SetBindings** abstract interface with the HANDLE of the named pipe over which the server has received the CPMSetBindingsIn message as its *QueryIdentifier* argument, with the **_hCursor** handle as its *CursorHandle* argument, and with the **aColumns** field as its Columns argument. The server MUST report any error code returned as the *Error* output parameter.
5. Respond to the client with a message header (only), with **_msg** set to CPMSetBindingsIn and **_status** set to the results of the specified binding.
6. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:
   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.9 Receiving a CPMGetNotify Request

When the server receives a CPMGetNotify message from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMGetNotify message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.<33>
2. Call the **GetQueryStatusChanges** abstract interface with the HANDLE of the named pipe over which the server has received the CPMGetNotify message as its *QueryIdentifier* argument. If the

ChangesPresent output parameter is true, then the server MUST respond with a CPMSendNotifyOut message and MUST set the **_watchNotify** field of this message to the *LatestChange* parameter value that is output from the call.

3. At a later time, if there is a change in the query results set as determined by polling the GSS with **GetQueryStatusChanges** interface calls, the server MUST send exactly one CPMSendNotifyOut message to the client and MUST specify the change in the **_watchNotify** field.

4. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.10 Receiving a CPMGetApproximatePositionIn Request

When the server receives a CPMGetApproximatePositionIn message request from the client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMGetApproximatePositionIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Call the **ClientQueryHasCursorHandle** abstract interface to the GSS with the HANDLE of the named pipe over which the server has received the CPMGetApproximatePositionIn message as its *QueryIdentifier* argument and with the **_hCursor** handle as its *CursorHandle* argument. If the *ContainsHandle* output parameter is not true, the server MUST report an E_FAIL (0x80004005) error.

3. Call the **GetApproximatePosition** abstract interface with the HANDLE of the named pipe over which the server has received the CPMGetApproximatePositionIn message as its *QueryIdentifier* argument, with the **_hCursor** handle as its *CursorHandle* argument, and with **_bmk** as its *Bmk* argument. Use the *riRowBmk* output parameter as the numerator and use the total number of rows for this cursor as the denominator for the CPMGetApproximatePositionOut message. The total number of rows is the value of the *rcRowsTotal* output parameter from a **GetExpensiveProperties** abstract interface call, using *QueryIdentifier* and **_hCursor** as arguments to the call.

   Note If this step fails for any reason, the server MUST report any error code encountered in performing the request in accordance with Win32 Error Codes in [MS-ERREF].

4. Respond to the client with a CPMGetApproximatePositionOut message.

5. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.11 Receiving a CPMCompareBmkIn Request

When the server receives a CPMCompareBmkIn message request from the client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMCompareBmkIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Call the **ClientQueryHasCursorHandle** abstract interface to the GSS with the HANDLE of the named pipe over which the server has received the CPMCompareBmkIn message as its *QueryIdentifier* argument and with the **_hCursor** handle as its *CursorHandle* argument. If the *ContainsHandle* output parameter is not true, the server MUST report an E_FAIL (0x80004005) error.

3. Prepare a CPMCompareBmkOut message.

    - If the bookmark handles are equal, **_dwComparison** MUST be set to DBCOMPARE_EQ.

    - Otherwise, the server MUST do the following:

        - Find rows that are referred to by each bookmark handle in the query results: the server MUST retrieve the indices of both rows within the rowset by calling the **GetBookmarkPosition** abstract interface twice, with the HANDLE of the named pipe over which the server has received the CPMCompareBmkIn message as its *QueryIdentifier* argument, with the **_hCursor** handle as its *CursorHandle* argument, and the **bmkFirst** and **bmkSecond** bookmark handles, respectively as the *bmkHandle* argument.

        - If the chapter handle in CPMCompareBmkIn is invalid, or if one or both of the rows are not in the given chapter, the behavior is undefined.

        - Otherwise, when both rows are in the same chapter, the server MUST then compare the position values obtained via a **GetBookmarkPosition** interface call and set **_dwComparison** to DBCOMPARE_LT if the position of the first row is smaller than the position of the second row; otherwise, **_dwComparison** MUST be set to DBCOMPARE_GT.

4. Respond to the client with the configured CPMCompareBmkOut message.

5. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

    - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.

    - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

    Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.12 Receiving a CPMRestartPositionIn Request

When the server receives the CPMRestartPositionIn message request from the client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMRestartPositionIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Call the **ClientQueryHasCursorHandle** abstract interface to the GSS with the HANDLE of the named pipe over which the server has received the CPMRestartPositionIn message as its *QueryIdentifier* argument, and with the **_hCursor** handle as its *CursorHandle* argument. If the

ContainsHandle output parameter is not true, the server MUST report an E_FAIL (0x80004005) error.

3. Move the cursor to the beginning of the chapter identified by the chapter handle by calling the **SetNextGetRowsPosition** abstract interface with the HANDLE of the named pipe over which the server has received the CPMRestartPositionIn message as its *QueryIdentifier* argument, with the **_hCursor** handle as its *CursorHandle* argument, and with the **_chapt** handle as the *chapter* argument. When the chapter handle is DB_NULL_HCHAPTER, the corresponding chapter is the main rowset of the query.

   Note If this step fails for any reason, the server MUST report any error code encountered in performing the request in accordance with Win32 Error Codes in [MS-ERREF].

4. Respond to the client with a CPMRestartPositionIn message.

5. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.13 Receiving a CPMFreeCursorIn Request

When the server receives a CPMFreeCursorIn message request from the client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMFreeCursorIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Call the **ClientQueryHasCursorHandle** abstract interface to the GSS with the HANDLE of the named pipe over which the server has received the CPMFreeCursorIn message as its *QueryIdentifier* argument and with the **_hCursor** handle as its *CursorHandle* argument. If the *ContainsHandle* output parameter is not true, the server MUST report an error.<34>

3. Release the cursor and associated resources for this cursor handle by calling the **ReleaseCursor** abstract interface with the HANDLE of the named pipe over which the server has received the CPMFreeCursorIn message as its *QueryIdentifier* argument and with the **_hCursor** handle as its *CursorHandle* argument.

4. Respond with a CPMFreeCursorOut message, setting the **_cCursorsRemaining** field to the number of cursors remaining in this client's list, as returned in the *NumCursorsRemaining* output parameter to the **ReleaseCursor** abstract interface call made in the previous step.

5. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.14 Receiving a CPMDisconnect Request

When the server receives a CPMDisconnect message request from the client, the server MUST do the following:

- Remove the HANDLE of the named pipe over which the server has received the CPMDisconnect message from the **ConnectedClientsIdentifiers** list.
- Remove the corresponding entry from the **ConnectedClientVersions** list.
- Call the **ReleaseQuery** abstract interface to the GSS with the HANDLE of the named pipe over which the server has received the CPMDisconnect message as its *QueryIdentifier* argument.

3.1.5.2.15 Receiving a CPMFindIndicesIn Request

When the server receives a CPMFindIndicesIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMFindIndicesIn message. If it is not present, the server MUST report an E_INVALIDARG (0x80070057) error.
2. Prepare a CPMFindIndicesOut message. If this step fails for any reason, the server MUST report any error code encountered in performing the request in accordance with Win32 Error Codes in [MS-ERREF].
3. The CPMFindIndicesOut message MUST contain one hierarchical group coordinate to the next occurrence of one of the document identifiers specified in the CPMFindIndicesIn message or, if no such occurrence was found, the server MUST indicate this by setting **_cDepthNext** to zero. In the case of a non-grouping query, **_cDepthNext** will be 1, indicating a single offset into the current rowset.

   The next occurrence is found by calling the **FindNextOccurrenceIndex** abstract interface to the GSS, with the HANDLE of the named pipe over which the server has received the CPMFindIndicesIn message, **_prgiRowPrev** as the previous coordinates list (or NULL if **_cDepthPrev** is zero), and array **_pwids[0]** as arguments.

   Note Currently, the server only supports a single document identifier lookup and discounts any workids in **_pwids** other than the first one. The next occurrence coordinate list is returned in the *NextOccCoordinatesList* output parameter, or if not found, then the *NextOccExists* parameter is not true.
4. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.16 Receiving a CPMGetRowsetNotifyIn

When the server receives a CPMGetRowsetNotifyIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMGetRowsetNotifyIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Prepare a CPMGetRowsetNotifyOut message. If this step fails for any reason, the server MUST report any error code encountered in performing the request in accordance with Win32 Error Codes in [MS-ERREF].

3. The server MUST make a call to the **GetLastUnretrievedEvent** abstract interface of the GSS with the HANDLE of the named pipe over which the server has received the CPMGetRowsetNotifyOut message as its *QueryIdentifier* argument and populate the CPMGetRowsetNotifyOut message with the corresponding output parameters.

4. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.17 Receiving a CPMGetScopeStatisticsIn

When the server receives a CPMGetScopeStatisticsIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMGetScopeStatisticsIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Prepare a CPMGetScopeStatisticsOut message. If this step fails for any reason, the server MUST report any error code encountered in performing the request in accordance with Win32 Error Codes in [MS-ERREF].

3. Call the **GetQueryStatistics** abstract interface of the GSS, with the HANDLE of the named pipe over which the server has received the CPMGetScopeStatisticsIn message as the *QueryIdentifier* argument for the call. Populate the CPMGetScopeStatisticsOut message fields with the corresponding output arguments from this call.

   Note The server SHOULD return zero for all statistics unless the client has explicitly requested their availability by setting the DBPROP_ENABLEROWSETEVENTS property to TRUE.

4. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.
   - E_ACCESSDENIED: generated when the client does not have permissions to access a needed resource such as a file result or a catalog.

   Any other error code can be returned, but it will be treated as informative only.

3.1.5.2.18 Receiving a CPMSetScopePrioritizationIn

This message is currently implemented only in the Windows 7 operating system. If the server fails implementation of this message, it can report a STATUS_INVALID_PARAMETER error. Otherwise, when the server receives a CPMSetScopePrioritizationIn message request from a client, the server MUST do the following:

1. Search the **ConnectedClientsIdentifiers** list for the HANDLE of the named pipe over which the server has received the CPMSetScopePrioritizationIn message. If it is not present, the server MUST report a STATUS_INVALID_PARAMETER (0xC000000D) error.

2. Prepare a CPMSetScopePrioritizationOut message. If this step fails for any reason, the server MUST report any error code encountered in performing the request in accordance with Win32 Error Codes in [MS-ERREF].

3. Call the **SetScopePriority** abstract interface of the GSS, with the HANDLE of the named pipe over which the server has received the CPMSetScopePrioritizationIn message and the **priority** field as arguments.

4. If **eventFrequency** is not zero, start an EventTimer timer that expires after an interval defined by **eventFrequency**, in milliseconds. Otherwise, if **eventFrequency** is zero, then call the **FilterOutScopeStatisticsMessages** abstract interface of the GSS, with the HANDLE of the named pipe over which the server has received the CPMSetScopePrioritizationIn message as the argument for the call.

5. The CPMSetScopePrioritizationOut message MUST simply acknowledge successful receipt of the CPMSetScopePrioritizationIn message and that the **SetScopePriority** abstract interface has been called, with the HANDLE of the named pipe over which the server has received the CPMSetScopePrioritizationIn message and the **priority** field as arguments to the call.

6. Report any errors encountered during message preparation or during any abstract interface call to the GSS. Errors that are specific to this request:

   - E_OUTOFMEMORY: generated by any resource allocation failure on the server or service side.
   - STATUS_INVALID_PARAMETER: generated when any of the parameters passed in by the client is invalid. Invalid parameters are those that do not obey the corresponding data structure layout as defined for their types in this document.

   Any other error code can be returned, but it will be treated as informative only.

3.1.6 Timer Events

The following timer is required.

EventTimer: The timer event for this timer calls the **GenerateScopeStatisticsEvent** abstract interface of the GSS, with the *QueryIdentifier* value associated to this EventTimer as its argument. This does not result in any message being sent to the client. It simply ensures that the GSS generates a Scope Statistics event which can later be retrieved by the client via a CPMGetRowsetNotifyIn message.

3.1.7 Other Local Events

This section describes the use of several abstract interfaces.

IsCatalogAvailable

This abstract interface is used to determine whether the specified catalog is available for querying.

Inputs: **CatalogName**, a VT_BSTR, the name of the desired catalog.

Outputs: **IsAvailable**, a Boolean, true if and only if all of the following conditions are met.

- The catalog name matches a supported catalog implemented by the service (case-insensitive). All versions of WSS covered in this document support only the catalog with name "Windows\SYSTEMINDEX".
- The Generic Search Service (GSS) is currently able to return results for queries. This is implementation-specific and can be used by other implementations, for example, to communicate the temporary unavailability of the server during initialization.

GetServerVersions

This abstract interface is used to retrieve the Generic Search Service (GSS) version, as well as version information about the operating system and the natural language components on the operating system on which GSS runs.

Inputs: None.

Outputs: DWORD **dwWinVerMajor**, DWORD **dwWinVerMinor**, DWORD **dwNLSVerMajor**, DWORD **dwNLSVerMinor**, DWORD **serverVersion**, Boolean **supportsVersioningInfo**.

Constraints: If the server supports reporting versioning information, return **dwWinVerMajor**, **dwWinVerMinor**, **dwNLSVerMajor**, and **dwNLSVerMinor** according to section 3.1.5.2.1, and set **supportsVersioningInfo** to true. Otherwise, set **supportsVersioningInfo** to false. In both cases, **serverVersion** MUST be returned.

GetState

This abstract interface returns information about the Generic Search Service (GSS).

Inputs: None.

Outputs: The parameters of a CPMCiStateInOut message, as described in section 2.2.3.1.

Constraints: None.

StoreClientInformation

This abstract interface is used to communicate the client information to the GSS upon client connection via a CPMConnectIn (section 2.2.3.2) message. This information will be used later for access checks on query results for this client. This information is released upon the call to **ReleaseQuery**.

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **ConnectMessage**, a CPMConnectIn message.
- **NamedPipeHandle**, the named pipe handle over which the client has connected.

Outputs: None.

RunNewQuery

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- Query information:

- **ProjectionColumnsOffsets**: A CcolumnSet structure containing the property offsets for properties in CpidMapper that are returned as columns. For example, if **RunNewQuery** was being generated from SQL, these columns would correspond to the arguments of the SELECT statement.
- **RestrictionSet**: A CrestrictionArray structure containing the command tree of the query. In the case of a multi-level grouping query, these restrictions apply only to the filtered set found at the leaf level.
- **SortOrders**: A **CsortSet** structure indicating the sort order of the query. The Sort Set also applies only to the set of results found at the leaf level, in the case of a multi-level grouping query.
- **Groupings**: A **CcategorizationSet** structure that contains the groups for the query, aggregate properties to return and any sort orders for non-leaf levels in the hierarchical rowset.
- **RowSetProperties**: A CrowsetProperties structure providing configuration information for the query.
- **PidMapper**: A CpidMapper structure that maps from property offsets to full property descriptions.
- **GroupArray**: A CcolumnGroupArray structure, describing property weights for probabilistic ranking.
- **Lcid**: A 32-bit unsigned integer, indicating the user's locale for this query, as specified in [MS-LCID].

Outputs:

- **CanRunQueryNow**: Boolean, true if and only if another query is not already in progress. A query is defined to be in progress at any time between (a) and (b):
  - Any time when **RunNewQuery** was called with **QueryIdentifier** as an argument and returned true and **QueryParametersError** (see following) was returned as zero.
  - The first time **ReleaseQuery** was called with **QueryIdentifier** as an argument after (a) and returned true.
- **QueryParametersError**: This is a DWORD set either to 0 for success or one of the values below:

Value	Description
QUERY_E_ALLNOISE 0x80041605	The query contained only ignored words.
QUERY_E_DIR_ON_REMOVABLE_DRIVE 0x8004160B	Specified directory is on a removable medium.
QUERY_E_DUPLICATE_OUTPUT_COLUMN 0x80041608	One or more columns in the output column list are duplicate.
QUERY_E_FAILED 0x80041600	Call failed for unknown reason.
QUERY_E_INVALIDQUERY 0x80041601	Invalid parameter.
QUERY_E_INVALIDRESTRICTION	The query restriction could not be parsed.

Value	Description
0x80041602	
QUERY_E_TIMEDOUT 0x80041607	The query exceeded its execution time limit. The time limit is defined by the **cCmdTimeout** property in the **RowSetProperties** argument.
QUERY_E_TOOCOMPLEX 0x80041606	The query was too complex to be executed. Normally this refers to the maximum number of nodes in the restriction tree. Currently, the WSS maximum is set in the Windows Registry to 520,000.
QUERY_S_NO_QUERY 0x8004160C	The catalog is in a state where indexing continues, but queries are not allowed.

- **CursorHandlesList**: If **QueryParametersError** is 0, return a list of cursor handles, otherwise return an empty list.
- **fTrueSequential**: A Boolean. In GSS, its meaning is as described in CPMCreateQueryOut (section 2.2.3.5).<35>
- **fWorkidUnique**: A Boolean. This is set to true unless the GSS is unable to return results.

Constraints:

- The cursor handles in the returned **CursorHandlesList** MUST be different from each other.
- The number of cursor handles returned is equal to the number of categories in the **Groupings** parameter and every returned handle identifies the results in the same-index category in the **Groupings** parameter. These handles will be referenced in later invocations of abstract interfaces such as **GetRows**.
- For the query provided in the CPMCreateQueryIn message, the server can use the inverted index in such a way that query results will likely be delivered faster. Otherwise, there would be a greater latency in delivering query results.
- fWorkidUnique: This is set to true as long as the GSS is able to return results.
- GroupArray constraint: Any CcolumnGroup present in GroupArray MUST have the groupPid value high bits set to 0x7fff. In other words☹0xFFFF0000 & groupPid) == 0x7FFF0000. Moreover, the low bits value: 0x0000ffff || groupPid MUST represent the index of the group property entry in the PidMapper aPropSpec array. The abovementioned CfullPropspec entry in the PidMapper aPropSpec array MUST be identical to the _Property CfullPropspec of any CprobRestriction in the **RestrictionSet** that references this CcolumnGroup.

ClientQueryHasCursorHandle

This abstract interface tests whether a client query has a given cursor handle associated.

Inputs:

- **QueryIdentifier**: A 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**: A 32-bit unsigned integer.

Outputs:

- **ContainsHandle**: A Boolean, true if and only if the last **RunNewQuery** call that didn't return an error and that had **QueryIdentifier** as its argument returned **CursorHandle** as one of the **CursorHandlesList** handles.

Constraints:

- If **ReleaseCursor** has been called with the **QueryIdentifier** and **CursorHandle** parameters, then **ContainsHandle** MUST be set to false.

GetQueryStatus

This abstract interface retrieves information about the status of the query.

Inputs:

- **QueryIdentifier**: A 32-bit unsigned integer. This value uniquely identifies a query to this server.

Outputs:

- **QueryStatus**: A 32-bit unsigned integer.<36>
- **Error**: An HRESULT set to 0 if no error was encountered. Otherwise, this value is set to any error code encountered in performing the request in accordance with Win32 Error Codes in [MS-ERREF].

GetRatioFinishedParams

This abstract interface retrieves information about the current state of query processing.

Inputs:

- **QueryIdentifier**: A 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**: A 32-bit unsigned integer identifying the rowset of interest in the query identified by the **QueryIdentifier**.

Outputs:

- **rdwRatioFinishedDenominator**: A 32-bit unsigned integer identifying the number of currently computed results that satisfy the **RestrictionSet** passed in the last successful call to **RunNewQuery** with the same **QueryIdentifier** argument. In WSS, the number of computed results satisfying the **RestrictionSet** is known before the SortOrders + Groupings. This value represents the total number of results for the query.
- **rdwRatioFinishedNumerator**: A 32-bit unsigned integer identifying the number of currently computed results that satisfy both the **RestrictionSet** and the SortOrders + Groupings passed in the last successful call to **RunNewQuery** with the same **QueryIdentifier** argument. In WSS, this number can differ from rdwRatioFinishedDenominator because it is possible to compute the denominator before calculating the SortOrders + Groupings. The **rdwRatioFinishedNumerator** represents the number of results that have currently been computed with the correct SortOrders + Groupings.
- **cRows**: A 32-bit unsigned integer indicating the total number of results for the query. In WSS versions after Windows Search 4.0, this value was trivially set to zero because this output parameter is no longer used by any Windows clients.
- **fNewRows**: A Boolean value that is true if and only if there are more results than have already been requested via the **GetRows** abstract interface. In WSS versions greater than or equal to Windows Search 4.0, this is true if and only if **rdwRatioFinishedDenominator** != **rdwRatioFinishedNumerator**.

Constraints:

- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, the behavior of **GetRatioFinishedParams** is undefined.

- **rdwRatioFinishedDenominator** MUST be greater than or equal to **rdwRatioFinishedNumerator**.
- **rdwRatioFinishedNumerator** MUST be greater than or equal to the maximum index of any row returned via the **GetRows** abstract interface called with **QueryIdentifier** and **CursorHandle** arguments.
- **rdwRatioFinishedDenominator** MUST be smaller than or equal to **rcRowsTotal** returned by calling the **GetExpensiveProperties** abstract interface called with **QueryIdentifier** and **CursorHandle** arguments.
- **cRows** has the same semantics and has to have an identical value to that of **rcRowsTotal** returned by calling the **GetExpensiveProperties** abstract interface called with **QueryIdentifier** and **CursorHandle** arguments.

GetApproximatePosition

This abstract interface retrieves the approximate position of a bookmark within a rowset.

Inputs:

- **QueryIdentifier**: A 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**: A 32-bit unsigned integer identifying the rowset of interest in the query identified by the **QueryIdentifier**.
- **Bmk**: A 32-bit value indicating the handle of a bookmark whose position is to be retrieved.

Outputs:

- **riRowBmk**: A 32-bit unsigned integer identifying the approximate position of the bookmark in the rowset identified by **CursorHandle** for the query identified by **QueryIdentifier**.

Constraints:

- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, the behavior of **GetApproximatePosition** is undefined.

GetWhereid

This abstract interface retrieves the query identifier of a given query.<37>

Inputs:

- **QueryIdentifier**: A 32-bit unsigned integer. This value uniquely identifies a query to this server.

Outputs:

- **whereID**: A 32-bit unsigned integer that defines a unique WHEREID for referring to the **RestrictionSet** CRestrictionArray argument to the latest successful **RunNewQuery** abstract interface call. This restriction can be reused as a restriction in future calls to **RunNewQuery** as long as there is still a cursor corresponding to the cursor handles returned by the latest successful call to **RunNewQuery** that has not been freed using **ReleaseQuery**. This provides the GSS the option of sharing the evaluation of the restriction across queries.

Constraints:

- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract

interface has returned true. If this is not the case, then the behavior of **GetRatioFinishedParams** is undefined.

- The same value for whereID is provided during any subsequent calls to **GetWhereId** that have the same **QueryIdentifier** argument.

GetExpensiveProperties

This abstract interface retrieves expensive properties. Specifically, these are properties that are evaluated over the entirety of the result set and are thus expensive to compute.

Inputs:

- **QueryIdentifier**: A 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**: A 32-bit unsigned integer identifying the rowset of interest in the query identified by the **QueryIdentifier**.

Outputs:

- **rcRowsTotal**: A 32-bit unsigned integer specifying the total number of rows in the rowset.
- **rdwResultCount**: A 32-bit unsigned integer specifying the number of unique results returned in the rowset.
- **Maxrank**: A 32-bit unsigned integer specifying the maximum rank found in the rowset.

Constraints:

- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, then the behavior of **GetExpensiveProperties** is undefined.
- The GSS MUST NOT output any parameters if the *DONOTCOMPUTEEXPENSIVEPROPERTIES* property wasn't set in the **RowSetProperties** argument to the last successful **RunNewQuery** abstract interface call with the **QueryIdentifier** argument.
- **rcRowsTotal** MUST be greater than or equal to the maximum index of any row returned via the **GetRows** abstract interface called with **QueryIdentifier** and **CursorHandle** arguments at any time in the past or future.
- **rdwResultCount** MUST be smaller than **rcRowsTotal**.
- **Maxrank** MUST be greater than or equal to maximum rank of any row returned via the GetRows abstract interface called with **QueryIdentifier** and **CursorHandle** arguments at any time in the past or future.

HasBindings

This abstract interface informs the caller about whether the GSS has a set of bindings for the specified query and cursor.

Inputs:

- **QueryIdentifier**: A 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**: A 32-bit unsigned integer identifying the rowset of interest in the query identified by the QueryIdentifier.

Outputs:

- **hasBindings**: A Boolean, true if and only if the **SetBindings** abstract interface was called successfully with **QueryIdentifier** and **CursorHandle** as arguments.

Constraints:

- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, then the behavior of **GetRatioFinishedParams** is undefined.
- Bindings cannot be set for cursors identifying rowsets for non-leaf categorization levels.

GetBookmarkPosition

This abstract interface retrieves the position within a rowset of a given bookmark.

Inputs:

- **QueryIdentifier**: A 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**: A 32-bit unsigned integer identifying the rowset of interest in the query identified by the **QueryIdentifier**.
- **bmkHandle**: A 32-bit unsigned integer identifying the bookmark within this rowset.

Outputs:

- **bmkIndex**: A 32-bit unsigned integer identifying the position within the rowset of the bookmark provided.

Constraints:

- Stability: This abstract interface needs to provide the same **bmkIndex** if and only if its arguments: **QueryIdentifier**, **CursorHandle**, and **bmkHandle** are the same.
- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, then the behavior of this interface is undefined.

SetNextGetRowsPosition

This abstract interface allows the caller to instruct the GSS to store the index within the rowset where to return results from next via the **GetRows** abstract interface.

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**, a 32-bit unsigned integer identifying the rowset of interest in the query identified by the **QueryIdentifier**.
- **Chapter**, a chapter identifying the range of rows of interest within the rowset identified by **CursorHandle**.
- **Index**, a 32-bit unsigned integer identifying where the next row will be retrieved from.

Outputs: None.

Constraints:

- If no error occurred, then any subsequent call to **GetNextGetRowsPosition** with the **QueryIdentifier**, **CursorHandle** and **chapter** arguments will return the Index value.

- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, then the behavior of this interface is undefined.

GetNextGetRowsPosition

This abstract interface retrieves the index within the rowset where the GSS will return results from next via the **GetRows** abstract interface.

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**, a 32-bit unsigned integer identifying the rowset of interest in the query identified by the **QueryIdentifier**.
- **Chapter**, a chapter identifying the range of rows of interest within the rowset identified by CursorHandle.

Outputs:

- **Index**, a 32-bit unsigned integer identifying the location of the next row to be retrieved.

Constraints:

- See the first constraint under the **SetNextGetRowsPosition** abstract interface.
- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, then the behavior of this interface is undefined.

GetRows

This abstract interface returns rows requested for the specified query and cursor. The position from which it returns rows can be set through the **SetNextRowsPosition** abstract interface.

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**, a 32-bit unsigned integer identifying the rowset of interest in the query identified by the **QueryIdentifier**.
- **NumRowsRequested**, a 32-bit unsigned integer identifying the number of rows to be retrieved.
- **FetchForward**, a 32-bit unsigned integer identifying whether the rows are to be fetched in forward order or in reverse. (0x00000000 for forward, 0x00000001 for reverse)

Outputs:

- **RowsArray**, an array of rows, with properties consistent with the bindings set via the last successful **SetBindings** call. See the second constraint.
- **NoMoreRowsToReturn**, a Boolean, true if and only if there are no more results to return that satisfy the query associated with **NamedPipe** for the cursor associated with **CursorHandle**.
- **NumRowsReturned**, a 32-bit unsigned integer identifying the number of rows returned.
- **Error**, a 32-bit unsigned integer identifying the error HRESULT of any error incurred during row retrieval.

Constraints:

- Results MUST be stable with respect to latest position set in **SetNextGetRowsPosition**. Mathematically, these conditions need to be met:
  - for any 32-bit unsigned integer i satisfying 1 <= i <= **totalNumberOfResults**, calling **SetNextGetRowsPosition**(**QueryIdentifier**, **CursorHandle**, i), and then **GetRows**(**QueryIdentifier**, **CursorHandle**, 1, 0) MUST produce the same result at any time this query is active (a query is active between a successful **RunNewQuery** call and a **ReleaseQuery** call with the **QueryIdentifier** argument)
  - for any 32-bit unsigned integers i, j, k satisfying 1 <= i <= k <= j <= **totalNumberOfResults**, calling **SetNextGetRowsPosition**(**QueryIdentifier**, **CursorHandle**, i), and then **GetRows**(**QueryIdentifier**, **CursorHandle**, j-i+1, 0) MUST produce the same (k – i +1)th row as the first and only row obtained by calling **SetNextGetRowsPosition**(**QueryIdentifier**, **CursorHandle**, **k**), and then **GetRows**(**QueryIdentifier**, **CursorHandle**, 1, 0)
  - same stability conditions MUST hold true when fetching results in reverse (**FetchForward** = 0x00000001)
- Index must be updated by any retrieval: Any **GetNextGetRowsPosition** call with the same **NamedPipeHanlde** and **CursorHandle** arguments made after the **GetRows** call must reflect the updated index: its value is incremented by **NumRowsReturned** or, if **FetchForward** was set to 0x00000001, then minus-**NumRowsReturned**.
- Result consistence with bindings:
  - When fetching rows, the GSS MUST compare each selected column's property value type to the type that is specified in the client's current set of bindings as set in the **SetBindings** abstract interface call with the same **QueryIdentifier** and **CursorHandle** as arguments. If the type in the binding was not VT_VARIANT, the GSS MUST attempt to convert the column's property value to that type. Otherwise, if the DBPROP_USEEXTENDEDDBTYPES flag was set in the client's DBPROPSET_QUERYEXT property set, or if the column's property value was not a VT_VECTOR type, the property value MUST be returned in its normal type. If none of these are the case (that is, the server has a VT_VECTOR type, and the client does not support VT_Vector), the server MUST attempt to convert it to a VT_ARRAY type as follows:
    - VT_I8, VT_UI8, VT_FILETIME, and VT_CLSID array elements cannot be converted and instead fail.
    - VT_LPSTR and VT_LPWSTR array elements MUST be converted to VT_BSTR.
    - Array elements of all other types MUST remain unchanged.

    Aggregates are returned as individual columns. They are mostly simple types except for ByFrequency, First, DateRange, and RepresentativeOf which are returned as compound types.
    - ByFrequency: Returns a VT_VECTOR of 3 VT_VECTORS. The first vector contains up to N values (as specified in the **ulMaxNumToReturn** field in the CAggregSpec). Their types are those specified by the **idColumn** field in the CAggregSpec. The second vector contains the corresponding count for each value. Their types are numeric (VT_UI4). The third vector contains a representative document identifier for each unique value. Their types are also numeric (VT_UI4).
    - First: Returns a VT_VECTOR of VT_VARIANT values. The number of values is at most N (as specified in the corresponding CAggregSpec)
    - DateRange: Returns a VT_VECTOR containing 2 VT_FILETIME structures. They define the lower bound of the requested range and the upper bound, respectively.

        - RepresentativeOf: Returns a VT_VECTOR of two VT_VECTORS. The first one contains the representative property values as specified in the corresponding CAggregSpec. These returned values are of type VT_VARIANT. There are at most N such values, as specified in the **ulMaxNumToReturn** field in the CAggregSpec. The second vector contains a corresponding document identifier for each of the values in the first vector. Each of the documents denoted by the above identifiers has the value of the specified property equal to that of the corresponding value in the first vector. The document identifiers returned are of (VT_UI4) types.

- Results SHOULD satisfy the restrictions passed in the last successful **RunNewQuery** abstract interface call.

- Results retrieved from the same chapter as set in **SetNextGetRowsPosition** SHOULD satisfy the corresponding Groupings entry in the last **RunNewQuery** abstract interface call with the same **NamePipeHandle** argument. Specifically, the returned results SHOULD have the same property as defined by the above mentioned Groupings entry.

- If the query was a hierarchical grouping query, and if the retrieval in this **GetRows** call did not refer to the leaf level rowset, then one of the returned properties MUST be a chapter handle that identifies the range of rows in the next level rowset that have the same property as defined by the corresponding Grouping in the previous **RunNewQuery** call with the same **NamedPipeHanlde** argument.

- If the requested number of rows was larger than the numbers left between the Index (as returned by a **GetNextGetRowsPosition** call with the same **NamedPipeHandle** and **CursorHandle** arguments) and the end of the rowset, NoMoreRowsToReturn MUST be set to true, and only as many rows as are present MUST be returned.

- Every row returned by the GSS MUST be ACL checked:

  When returning a row, the GSS MUST take the document identifier field of that row and call the HasAccessToWorkid abstract interface with **QueryIdentifier** and **Workid** as arguments. If it does not return true, then the GSS MUST skip this row.

- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, then the behavior of this interface is undefined.

HasAccessToWorkid

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.

- **Workid**, a 32-bit unsigned integer representing the document ID identifying the document for which permissions need to be checked.

Outputs:

- **HasAccess**, a Boolean, set to true if and only if the client as identified by the information passed during the **StoreClientInformation** abstract interface call with the same **QueryIdentifier** argument has file system access to the file identified by Workid.

  Specifically, this is done based on the Windows security permissions of the user identified by the security credentials received via the named pipe connection corresponding to the **NamedPipeHandle** argument passed to the last call to **StoreClientInformation** with the **QueryIdentifier** argument.

HasAccessToProperty

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **PropSpec**, a **CFullPropSpec** structure specifying the property to determine access to

Outputs:

- **HasAccess**, a Boolean, set to true if and only if the client as identified by the information passed during the **StoreClientInformation** abstract interface call with the same **QueryIdentifier** argument has file system access to the property identified by **PropSpec**.

  The access check is the same as the one in **HasAccessToProperty**.

GetPropertyValueForWorkid

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **Workid**, a 32-bit unsigned integer representing the document ID identifying the document for which a property is to be fetched.
- **PropSpec**, a **CFullPropSpec** structure specifying the property to retrieve.

Outputs:

- **Property**, a SERIALIZEDPROPERTYVALUE structure containing the property specified in PropSpec for the document identified by Workid.
- **ValueExists**, a Boolean set to true if and only if the specified property exists on the server and if the client has access to it. This MUST be determined by calling the **HasAccessToProperty** abstract interface with **QueryIdentifier** and **PropSpec** as arguments.

Constraints:

- Consistency across calls: Any two calls to **GetPropertyValueForWorkid** with the same arguments must return the same result.
- Consistency with the properties returned from **GetRows**: any row returned as part of the **GetRows** abstract interface call with the same **QueryIdentifier** argument that has the workid field identical to Workid MUST have the same value of the property identified by **PropSpec** (if requested and present) as the one returned here as the **Property** output parameter.

SetBindings

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**, a 32-bit unsigned integer identifying the rowset of interest in the query identified by the **QueryIdentifier**.
- **Columns**, an array of CTableColumn structures describing the columns of a row in the rowset.

Outputs:

- **Error**, a 32-bit unsigned integer identifying any error for this operation. If this step fails for any reason, the GSS MUST report any error code encountered in performing the request in accordance with Win32 Error Codes in [MS-ERREF].

Constraints:

- Other abstract interfaces such as **GetRows** have documented constraints that refer to the Columns passed in this call.
- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, then the behavior of this interface is undefined.

GetQueryStatusChanges

This abstract interface provides information about any query status changes that have occurred since the last time it was called.<38>

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **CursorHandle**, a 32-bit unsigned integer identifying the rowset of interest in the query identified by the QueryIdentifier.

Outputs:

- **LatestChange**, a 32-bit unsigned integer representing the change to the query. It MUST be one of the following values.

Value	Meaning
DBWATCHNOTIFY_ROWSCHANGED 0x00000001	The number of rows in the query rowset has changed.
DBWATCHNOTIFY_QUERYDONE 0x00000002	The query has completed.
DBWATCHNOTIFY_QUERYREEXECUTED 0x00000003	The query has been executed again.

- **ChangesPresent**, a Boolean indicating whether there have been any changes in query status.

Constraints:

- If there were no changes in the query result set since the last **GetQueryStatusChanges** call with the same **NamedPipe** and **CursorHandle** arguments, or if this is the first **GetQueryStatusChanges** call for this query/cursor pair, set ChangesPresent to false.
- In the case of many changes to query results, DBWATCHNOTIFY_ROWSCHANGED takes priority (that is, if the query was performed, re-executed, and then the number of rows changed and the query was performed again, then the event reported would be DBWATCHNOTIFY_ROWSCHANGED).
- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, then the behavior of this interface is undefined.

ReleaseCursor

This interface instructs the GSS to release all resources associated with a cursor of a query.

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.

- **CursorHandle**, a 32-bit unsigned integer identifying the rowset of interest in the query identified by the QueryIdentifier.

Outputs:

- **NumCursorsRemaining**, a 32-bit unsigned integer representing the number of cursors remaining for this query.

Constraints:

- This abstract interface assumes that the caller has already called **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments, respectively, and that the abstract interface has returned true. If this is not the case, then the behavior of this interface is undefined.
- Any further calls to **ClientQueryHasCursorHandle** with the **QueryIdentifier** and **CursorHandle** arguments will return false.
- NumCursorsRemaining has an initial value equal to the number of cursor handles returned by the previous call to **RunNewQuery** with the **QueryIdentifier** argument. It gets decremented with every **ReleaseCursor** call with the **QueryIdentifier** argument.
- If the NumCursorsRemaining is zero, then the GSS MUST call **ReleaseQuery** with the **QueryIdentifier** argument.

ReleaseQuery

This interface instructs the GSS to release all resources associated with a query, including the information stored during **StoreClientInformation**.

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.

Outputs: None.

Constraints:

- The GSS MUST call **ReleaseCursor** with the **QueryIdentifier** argument for all cursor handles that were returned by the **RunNewQuery** that haven't already been released via **ReleaseCursor** with the **QueryIdentifier** argument.

FindNextOccurrenceIndex

This interface allows retrieval of the next occurrence of a document within the result set of a hierarchicaly grouped query.<39>

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **PrevOccCoordinatesList**, a list of 32-bit unsigned integer signifying the group coordinates of the previous occurrence of the desired document within the query results. This is NULL if the caller requests the first occurrence within the rowset of the specified document.
- **Workid**, the document identifier for the desired document.

Outputs:

- **NextOccExists**, a Boolean set to true if and only if a "next" occurrence exists.
- **NextOccCoordinatesList**, a list of 32-bit unsigned integers signifying the group coordinates of the next occurrence of the desired document within the query results.

Constraints:

- Coordinates start at "1" rather than "0".
- Example: Consider grouping the following 4 results on Author at the top level and Keywords at the second level:

```
file1.txt
Author: 'author1'
Keywords: 'key1'

file2.txt
Author: 'author1'
Keywords: 'key2'

file3.txt
Author: 'author1'
Keywords: 'key2'

file4.txt
Author: 'author2'
Keywords: 'key3'
```

That grouping generates the following groups:

```
Author = 'author1'
    Keywords = 'key1'
        file1.txt
    Keywords = 'key2'
        file2.txt
        file3.txt
Author = 'author2'
    Keywords = 'key2'
        file4.txt
```

Then, the hierarchical grouping coordinate of the 'file3.txt' result will be <1, 2, 2>. The first integer in the coordinate, '1', represents the first Author category (labeled 'author1'). The second integer, '2', represents the second Keywords category within the 'author1' top level group (labeled 'key2'). Finally, the third integer, '2', represents the second result in the 'key2' category within the top level 'author1' group.

GetLastUnretrievedEvent

This abstract interface retrieves the last event that hasn't been reported yet. Events are scoped only to the result set of the current query.<40>

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.

Outputs:

- **Wid**, a 32-bit unsigned integer containing the document identifier that the event is for. This value MUST be zero if eventType is PROPAGATE_NONE or PROPAGATE_ROWSET.
- **EventType**, a 7-bit unsigned integer that MUST be one of the following values, indicating the type of event this message represents.

Value	Meaning
PROPAGATE_NONE	This event indicates that there were no available rowset events waiting on the server.

Value	Meaning
0	
PROPAGATE_ADD 1	This event indicates that an item was added to the index that could be relevant to the query originating the rowset.
PROPAGATE_DELETE 2	This event indicates that an item was deleted from the index that could be relevant to the query originating the rowset.
PROPAGATE_MODIFY 3	This event indicates that an item was re-indexed that could be relevant to the query originating the rowset.
PROPAGATE_ROWSET 4	This event is a rowset specific notification whose meaning is interpreted by the rowsetEvent field of this message.

- **MoreEvents**, a single bit that is set to 1 only if there are additional rowset events remaining.
- **RowsetItemState**, an 8-bit unsigned integer that MUST be one of the following values if EventType is PROPAGATE_ADD, PROPAGATE_DELETE, or PROPAGATE_MODIFY. This number indicates the state of the document identifier specified by **Wid** within the originating rowset associated with the query identified by the **QueryIdentifier** argument. For other EventType values this value MUST be set to zero.

Value	Meaning
ROWSETEVENT_ITEMSTATE_NOTINROWSET 0	The document identifier specified by Wid MUST NOT have been contained within the originating rowset.
ROWSETEVENT_ITEMSTATE_INROWSET 1	The document identifier specified by wid MUST be contained within the originating rowset.
ROWSETEVENT_ITEMSTATE_UNKNOWN 2	The document identifier specified by wid's containment within the originating rowset is unknown to the GSS.

- **ChangedItemState**, an 8-bit unsigned integer that MUST be one of the following values if EventType is PROPAGATE_MODIFY. This number indicates the state of the document identifier specified by Wid within the originating rowset associated with the query identified by the **QueryIdentifier** argument if the same query were to be run again following the change. For other EventType values, this value MUST be set to zero.

Value	Meaning
ROWSETEVENT_ITEMSTATE_NOTINROWSET 0	The document identifier specified by Wid would NOT be contained within a subsequent query.
ROWSETEVENT_ITEMSTATE_INROWSET 1	The document identifier specified by **Wid** would be contained within a subsequent query.
ROWSETEVENT_ITEMSTATE_UNKNOWN 2	Whether or not the document identifier specified by wid would be contained within a subsequent query is unknown to the GSS.

- **RowsetEvent**, an 8-bit unsigned integer that MUST be one of the following values if EventType is PROPAGATE_ROWSET. This number indicates the type of rowset event that this message represents. For other EventType values, this value MUST be set to zero.

Value	Meaning
ROWSETEVENT_TYPE_DATAEXPIRED 0	The data backing the rowset is no longer valid. RowsetEventData1 and RowsetEventData2 MUST be set to zero.
ROWSETEVENT_TYPE_FOREGROUNDLOST 1	The rowset no longer has foreground priority and has been reverted to high priority. Items that apply to this query will be indexed at a decreased rate. See section 2.2.3.34 for meaning of foreground and high priority. RowsetEventData1 and RowsetEventData2 MUST be set to zero.
ROWSETEVENT_TYPE_SCOPESTATISTICS 2	The number of indexed items, number of items that need to be indexed, or number of items that need indexing has changed. RowsetEventData1's high 32 bits contain a 32-bit unsigned integer indicating the number of items that need to be indexed that could be relevant to the originating rowset. RowsetEventData1's low 32 bits contain a 32-bit unsigned integer indicating the number of items that need to be re-indexed that could be relevant to the originating rowset. RowsetEventData2's high 32 bits MUST be set to zero. RowsetEventData2's low 32 bits contain a 32-bit unsigned integer indicating the number of indexed items that could be relevant to the originating rowset.

- **RowsetEventData1**, a 64 bit unsigned number whose meaning is dependent on RowsetEvent.
- **RowsetEventData2**, a 64 bit unsigned number whose meaning is dependent on RowsetEvent.

Constraints:

- The GSS MUST indicate the type of event response in EventType. All output parameters MUST be zero when not contributing to the value specified by EventType. The following fields MUST be set based upon the following event types.
  - PROPAGATE_NONE—EventType MUST be set to this value to indicate that there are no events available. No other fields are relevant to this response.

    Note: This is the only response possible if the last call to **RunNewQuery** with the **QueryIdentifier** argument did not have eEnableRowsetEvents set in the **RowSetProperties** argument.
  - PROPAGATE_ADD— EventType MUST be set to this value to indicate that a new item was added to the index that could be relevant to the associated query (the query identified by the **NamedPipeHanlde** argument). **Wid** MUST be set to the document identifier for the newly added document. RowsetItemState MUST be set either to indicate if the item would be included in the associated query if the query were executed again, or to indicate that this is unknown. MoreEvents SHOULD be set to indicate whether there is another event of any non-PROPAGATE_NONE type immediately available.
  - PROPAGATE_DELETE— EventType MUST be set to this value to indicate that an existing item was deleted from the index that could be relevant to the associated query. Wid MUST be set to the document identifier for the deleted document. RowsetItemState MUST be set either to indicate whether the document identifier is contained within the rowset, or to indicate that this is unknown. MoreEvents SHOULD be set to indicate whether there is another event of any non-PROPAGATE_NONE type immediately available.
  - PROPAGATE_MODIFY— EventType MUST be set to this value to indicate that an existing item was re-indexed and this item could have been previously relevant to the associated query or could have become relevant to the associated query as a result of being re-indexed. Wid MUST

be set to the document identifier for the re-indexed document. rowsetItemState MUST be set either to indicate if the document identifier is contained within the rowset, or to indicate that this is unknown. ChangedItemState MUST be set to indicate if the item would be included in the associated query if the query were executed again or MUST be set to indicate that this is unknown. MoreEvents SHOULD be set to indicate if there is another event of any non PROPAGATE_NONE type immediately available.

- PROPAGATE_ROWSET— EventType MUST be set to this value when the event is a non-item based event. **Wid** MUST be zero and RowsetEvent SHOULD be set to a value indicating the type of rowset event. Other fields MUST be set based upon this type as follows. Any field not specified explicitly below MUST be set to zero.
  - ROWSETEVENT_TYPE_DATAEXPIRED—RowsetEvent MUST be set to this value when the server wants to indicate that the data associated with the backing query could no longer be valid.
  - ROWSETEVENT_TYPE_FOREGROUNDLOST— RowsetEvent MUST be set to this value when the server wants to indicate that the associated query's explicitly requested priority has been changed from FOREGROUND to HIGH via a **SetScopePriority** abstract interface call with the same **QueryIdentifier** argument.
  - ROWSETEVENT_TYPE_SCOPESTATISTICS— RowsetEvent MUST be set to this value when sending an update due to a change in the number of indexed items, number of items that need to be indexed, or number of items that need to be re-indexed when these items are relevant to the associated query. RowsetEventData1's high 32 bits MUST contain an unsigned integer indicating the number of items that need to be indexed that could be relevant to the associated query. RowsetEventData1's low 32 bits MUST contain an unsigned integer indicating the number of items that need to be re-indexed that could be relevant to the associated query. RowsetEventData2's high 32 bits MUST be zero. RowsetEventData2's low 32 bits MUST contain the number of indexed items that could be relevant to the originating rowset.

- The GSS SHOULD NOT cache events unless they have explicitly requested by having called the **RunNewQuery** abstract interface with the same **QueryIdentifier** argument and with DBPROP_ENABLEROWSETEVENTS property to TRUE in the **RowSetProperties** argument. If this hasn't been the case, this call to GetLastUnretrievedEvent MUST return a PROPOGATE_NONE event.
- If the **FilterOutScopeStatisticsMessages** was called with the **QueryIdentifier** argument, then the GSS MUST NOT return ROWSETEVENT_TYPE_SCOPESTATISTICS events.

GetQueryStatistics

This abstract interface retrieves information about results of the specified query.<41>

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.

Outputs:

- **NumIndexedItems**, a 32-bit unsigned integer containing the number of items that are currently indexed that are relevant to the originating query (as identified by the **QueryIdentifier** argument).
- **NumOutstandingAdds**, a 32-bit unsigned integer containing the number of items that have yet to be indexed that could be relevant to the originating query.
- **NumOutstandingModifies**, a 32-bit unsigned integer containing the number of items that need to be re-indexed and that are relevant to the originating query.

Constraints:

- All output arguments MUST be set to zero if their availability has not been explicitly requested by having called the **RunNewQuery** abstract interface with the same **QueryIdentifier** argument and with DBPROP_ENABLEROWSETEVENTS property to TRUE in the **RowSetProperties** argument.

SetScopePriority

This abstract interface instructs the GSS to prioritize indexing of items on a per query basis.<42>

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.
- **Priority**, a 32-bit unsigned integer containing the type of prioritization requested for documents that could be relevant to the originating query (as defined by **QueryIdentifier**).

Value	Meaning
PRIORITY_LEVEL_FOREGROUND 0	Process items that could be relevant to the originating query before others as quickly as possible.
PRIORITY_LEVEL_HIGH 1	Process items that could be relevant to the originating query before others at the normal rate.
PRIORITY_LEVEL_LOW 2	Process items that could be relevant to the originating query before others, but after any other prioritization requests at the normal rate.
PRIORITY_LEVEL_DEFAULT 3	Process items at the normal rate.

Outputs: None.

Constraints:

- As a result of this call, the GSS MUST:
  - Prioritize indexing and re-indexing items according to the Priority specified in the **Priority** argument.
  - Respond with a RowsetEvent ROWSETEVENT_TYPE_SCOPESTATISTICS on the when the **GetLastUnretrievedEvent** abstract interface is called with the **QueryIdentifier** argument and there has been a change to the statistics for the originating query.
- The GSS SHOULD NOT prioritize rowsets unless this has been explicitly requested by having called the **RunNewQuery** abstract interface with the same **QueryIdentifier** argument and with DBPROP_ENABLEROWSETEVENTS property to TRUE in the **RowSetProperties** argument.

FilterOutScopeStatisticsMessages

This abstract interface instructs the GSS against propagating events of type ROWSETEVENT_TYPE_SCOPESTATISTICS from **GetLastUnretrievedEvent** calls.

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.

Outputs: None.

Constraints:

- The GSS MUST NOT return ROWSETEVENT_TYPE_SCOPESTATISTICS events as part of any **GetLastUnretrievedEvent** abstract interface call made with the **QueryIdentifier** argument.

Inflect

This abstract interface retrieves a list of inflected words based on a passed-in string.

Inputs:

- **phrase**, a non-null-terminated Unicode string containing the word/phrase to inflect.

Outputs:

- **inflections**, an array of non-null-terminated Unicode strings representing inflected versions of the phrase argument.
- **inflectionsCount**, a 32-bit unsigned integer representing the count of the inflections array.

Constraints: None – this is left at the latitude of the implementation.

GenerateScopeStatisticsEvent

This abstract interface causes the GSS to generate an event if any new information about the result set is available. The types of new information are described in the **GetLastUnretrievedEvent** abstract interface to the GSS. **GetLastUnretrievedEvent** consumes events generated through the **GenerateScopeStatisticsEvent** abstract interface call and returns them to the user when called.

Inputs:

- **QueryIdentifier**, a 32-bit unsigned integer. This value uniquely identifies a query to this server.

Outputs: None.

Constraints:

- Events generated through this abstract interface to the GSS MUST be the same events that are later returned by the **GetLastUnretrievedEvent** abstract interface call.

3.2 Client Details

3.2.1 Abstract Data Model

The following section specifies data and state maintained by the Windows Search Protocol client. The data is provided to explain how the protocol behaves. This section does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

A client has the following abstract state:

State	Description
Last Message Sent	A copy of the last message sent to the server.
Current Property Value	A partial value of a "deferred" property, which is in the process of being retrieved.
Current Bytes Received	The number of bytes received for the Current Property Value so far.
Named Pipe Connection	A connection to the server.

3.2.2 Timers

None.

3.2.3 Initialization

No actions are taken until a higher-layer request is received.

3.2.4 Higher-Layer Triggered Events

When a request is received from a higher layer, the client MUST create a named pipe connection to the server as specified in section 2.1. If it is unable to do so, the higher-layer request MUST be failed. That is, in case of a failure to connect, the higher level MUST retry the connection.

A header MUST be pre-pended with fields set as specified in section 2.2.2.

For messages that are specified as requiring a nonzero checksum, the **_ulChecksum** value MUST be calculated as follows:

1. The contents of the message after the **_ulReserved2** field in the message header MUST be interpreted as a sequence of 32-bit integers. The client MUST calculate the sum of the numeric values given by these integers.

2. Calculate the bitwise XOR of this value with 0x59533959.

3. Subtract the value given by **_msg** from the value that results from the bitwise XOR.

3.2.4.1 Remote Windows Search Service Catalog Management

Each message is triggered by a request from the higher layer. There is no message sequence enforced by the client for the Windows Search Protocol message requests for remotely managing the catalog. However, the server will reply with success only if the client previously connected by means of a CPMConnectIn message.

3.2.4.1.1 Sending a CPMCiStateInOut Request

Typically, the higher layer asks the protocol client to send a CPMCiStateInOut message when it requires information about the GSS on the server.

When requested to send this message, the client MUST do the following:

1. Send a CPMCiStateInOut message to the server.

2. Wait to receive a CPMCiStateInOut message from the server, silently discarding all other messages.

3. Report the value of the **_status** field of the response, and if it was successful, report the informational structure back to the higher layer.

3.2.4.2 Remote Windows Search Service Catalog Query Messages

With the exceptions of CPMGetRowsIn/CPMGetRowsOut and CPMFetchValueIn/CPMFetchValueOut, there is a one-to-one relationship between the Windows Search Protocol messages and higher-layer requests. For the two exceptions previously mentioned, multiple messages can be generated by the client to either satisfy size requirements or to retrieve a complete property. The higher layer typically keeps track of all query-specific information (such as cursor handles opened, legal values for bookmark and chapter handles, and **_wid** values for deferred property values) and also tracks if the client is in a connected state, but this is not enforced in any way by the client.

For illustrative purposes, the client portion of the diagram in section 3 illustrates this sequence for a simple GSS query.

3.2.4.2.1 Sending a CPMConnectIn Request

This message is typically the very first request from the higher layer. The higher level provides the protocol client with the information necessary to connect.

To serve the higher layer, the client MUST do the following:

1. Fill in the message, using the information provided by the higher-layer client (see section 2.2.3.2) in **_iClientVersion**, MachineName, UserName, PropertySet1, PropertySet2, and aPropertySet.
2. Set **_fClientIsRemote**, **_cbBlob**, **_cbBlob2**, **cPropSet**, and **cExtPropSet**, as specified in section 2.2.3.2.
3. Set the checksum in the **_ulChecksum** field.
4. Send the CPMConnectIn message to the server.
5. Wait to receive a CPMConnectOut message back from the server, silently discarding all other messages.
6. Report the value of the **_status** field of the response and, if it was successful, report the value of the **_serverVersion** back to the higher layer.
7. Compare 4 DWORDs past the **_serverVersion** field with the 4 DWORDs at the same offset of CPMConnectIn message. If the value of the DWORDs is different, assume that the server supports version information and decode **dwWinVerMajor**, **dwWinVerMinor**, **dwNLSVerMajor**, **dwNLSVerMinor** fields. If requested, the values or the information that the server does not support versioning MAY be reported to a higher layer.

For informative purposes, it is expected that higher layers will typically do the following actions upon successful connection, but these are not enforced by the Windows Search Protocol client:

- Use remote GSS catalog management messages for administrative tasks.
- Use a CPMCreateQueryIn request to create a search query for the purpose of retrieving results from the catalog.

3.2.4.2.2 Sending a CPMCreateQueryIn Request

The higher layer will typically provide the information for the query creation after the protocol client is connected. The higher layer provides the client with a restrictions set, column set, sort order rules, and categorization set (each of which can be omitted), rowset properties, and property ID mapper structure.

When this request is received from a higher layer, the client MUST perform the following:

1. Prepare a CPMCreateQueryIn as follows.
    - If a columns set is present, set CColumnsSetPresent to 0x01 and fill the **ColumnsSet** field.
    - If restrictions are present, set CRestrictionPresent to 0x01 and fill the **Restriction** field.
    - If a sort set is present, fill the **SortSet** field.
    - If a categorization set is present, set CSortSetPresent to 0x01 and fill the **CategorizationSet** field.
    - Set the rest of fields as specified in section 2.2.3.4.

2. Calculate **_ulCheckSum** field in the header.

3. Send the CPMCreateQueryIn message to the server.

4. Wait to receive the CPMCreateQueryOut message, silently discarding all other messages.

5. Report the value of the **_status** field of the response and, if it was successful, report the array of cursor handles and informative Boolean values (as specified in section 2.2.3.5) back to the higher layer.

3.2.4.2.3 Sending a CPMSetBindingsIn Request

Typically, the higher layer will set bindings for each column to be returned in the rows when it already has a valid cursor handle (after successfully receiving CPMCreateQueryOut). The higher layer is expected to provide an array of CTableColumn structures for the **aColumns** field and a valid cursor handle.

When this request is received from the higher layer, the client MUST perform the following:

1. Calculate the number of CTableColumn structures in the **aColumns** array, and set the **cColumns** field to this value.

2. Calculate the total size in bytes of the **cColumns** and **aColumns** fields, and set the **_cbBindingDesc** field to this value.

3. Set the specified fields in the CPMSetBindingsIn message to the values provided by the higher application layer. Set the **_ulChecksum** field to the value calculated, as specified in section 3.2.5.

4. Send the completed CPMSetBindingsIn message to the server.

5. Wait to receive a CPMSetBindingsIn message from the server, discarding all other messages.

6. Indicate the status from the **_status** field of the response to the higher layer.

For informative purposes, it is expected that higher layers will typically request a client to send a CPMGetRowsIn message, but this is not enforced by the Windows Search Protocol.

3.2.4.2.4 Sending a CPMGetRowsIn Request

When the higher layer is about to receive rows information, it will provide the protocol client with valid cursor and chapter handles and give an appropriate seek description. Typically, a higher layer is expected to do so when it has a valid cursor and/or chapter handle, and the bindings have been set with the CPMSetBindingsIn message. To access the rowset in a chapter, the higher layer is to use the chapter handle received from the server in a previous CPMGetRowsOut message.

When this request is received from the higher layer, the client MUST perform the following:

1. Determine what unsigned integer value to specify for the **_cbReadBuffer** field. To determine this value, the client MUST take the maximum value from the following:

   - One thousand times the value of the **c_RowsToTransfer** field.

   - The value of **_cbRowWidth**, rounded up to the nearest 512-byte multiple.

   - Take the higher of these two values, up to the 16-KB limit.

   - In cases where a single row is larger than 16 KB, the server cannot return results to this query.

2. Specify a client base for variable-sized row data in the client address space in the **_ulClientBase** field.<43>

3. Calculate the size of the seek description and set it in the **_cbSeek** field.

4. Set the value of **cbReserved** (which would act as an offset for the start of the rows contained in the **Rows** field in the CPMGetRowsOut message) to the value of **_cbSeek** plus 0x14.

5. Send a CPMGetRowsIn message to the server.

3.2.4.2.5 Sending a CPMFetchValueIn Request

If the client receives a CPMGetRowsOut response from the server with the column's **Status** field set to StoreStatusDeferred (0x01), it means that the property value was not included in the **Rows** field of the CPMGetRowsOut message. In this case, the higher layer typically requests that the protocol client retrieve the value by means of a CPMFetchValueIn message, and provides the PropSpec and **_wid** value for a deferred property, which the protocol client MUST use in the first CPMFetchValueIn message.

If this is the first CPMFetchValueIn message the client has sent to request the specified property, the client MUST perform the following:

1. Set all the fields in a message, as specified in section 2.2.3.15.

2. Set _cbSoFar to 0x00000000.

3. Set current bytes received to 0.

4. Send the CPMFetchValueIn message to the server.

3.2.4.2.6 Sending a CPMFreeCursorIn Request

After the higher level is no longer using the search query, it can release the resources on the server by requesting that the client send a CPMFreeCursorIn message.

When this request is received, the client MUST send a CPMFreeCursorIn message to the server containing the handle specified by the upper layer.

3.2.4.2.7 Sending a CPMDisconnect Message

If the higher layer has no more queries for the Generic Search service (GSS), to free up more server resources, the application can request that the client send a CPMDisconnect message to the server. When this query is received, the client MUST simply send the message as requested. There is no response to this message from the server.

3.2.4.2.8 Sending a CPMFindIndicesIn Request

When the higher layer is about to request document identifier location within a rowset information, it will provide the protocol client with a set of document identifiers to look for. The higher layer can also provide a previous hierarchical group coordinate indicating where to start searching. The higher layer can use the coordinate returned in a previous CPMFindIndicesOut message for this purpose.

When this request is received from the higher layer, the client MUST perform the following:

1. Ensure there is at least one document identifier to search for. This is enforced by setting **_cWids** greater than zero.

2. Write exactly **_cWids** document identifiers to the **_pwids** array.

3. Write the coordinate retrieved from the higher layer to the **_prgiRowPrev** array and write its size to the **_cDepthPrev** value. If no such coordinate is available, the client MUST set **_cDepthPrev** to zero.

4. Send the CPMFindIndicesIn message to the server.

3.2.4.2.9 Sending a CPMGetRowsetNotifyIn Request

If the higher layer requires rowset eventing, it can send a CPMGetRowsetNotifyIn message to the server to request the next available event at any timed interval it chooses.

3.2.4.2.10 Sending a CPMGetScopeStatisticsIn Request

When the higher layer wants to retrieve statistics regarding the number of indexed items, number of outstanding items to be indexed, or number of items needing re-indexed that are relevant to its query, it can send a CPMGetScopeStatisticsIn message to the server to request these values at any interval it chooses.

3.2.4.2.11 Sending a CPMSetScopePrioritizationIn Request

When the higher layer wants to modify the indexing priority of documents that could be relevant to its query, it will send a CPMSetScopePrioritizationIn message to the server to request that the priority of items relevant to this query be modified.

When this request is received form the higher layer, the client MUST perform the following:

1. Set the priority in the CPMSetScopePrioritizationIn message to the priority requested by the higher layer.

2. Set the eventFrequency in the CPMSetScopePrioritizationIn message to the frequency requested by the higher layer, or to zero if the priority requested is PRIORITY_LEVEL_DEFAULT.

3. Send the CPMSetScopePrioritizationIn message to the server.

3.2.5 Processing and Sequencing Rules

When the client receives a message response from the server, the client MUST use the Last Sent Message to determine if the message received from the server is the one expected by the client. All messages with the **_msg** field different from that in the Last Sent Message MUST be ignored.

3.2.5.1 Receiving a CPMCreateQueryOut Response

When the client receives a CPMCreateQueryOut message response from the server, the client MUST return **_status**, and if the status is successful, return the cursor handle values back to the higher layer. Any further actions are determined by the higher layer.

Because the higher layer can detect the query structure, it is expected that the correct number of cursor handles will be returned in the CPMCreateQueryOut message. The cursor handles are returned in the following order: the first handle is to the unchaptered rowset and the second handle is to the first chaptered rowset (which is the grouping of results based on the first category specified in the **CategorizationSet** field of the CPMCreateQueryIn message).

For informative purposes, it is expected that higher layers can perform the following actions, but these are not enforced by the Windows Search Protocol client:

- Use CPMSetBindingsIn to set the bindings for individual columns and perform any subsequent actions on query the path.

- Use CPMGetQueryStatusIn to check on the execution progress of a query.

- Use CPMRatioFinishedIn to request the completion percentage of a query.

3.2.5.2 Receiving a CPMGetRowsOut Response

When the client receives a CPMGetRowsOut message response from the server, the client MUST perform the following:

1. Check whether the **_status** field in the header indicates success or failure.
2. If the **_status** value is STATUS_BUFFER_TOO_SMALL (0xC0000023), the client MUST check the Last Message Sent state. If it does not contain a CPMGetRowsIn message, the received message MUST be silently ignored. Otherwise the client MUST send to the server a new CPMGetRowsIn message with all fields identical to the stored one, except that the **_cbReadBuffer** MUST be increased by 512 (but not greater than 0x4000). If **_status** is STATUS_BUFFER_TOO_SMALL (0xC0000023), and the Last Message Sent already has **_cbReadBuffer** equal to 0x4000, the client MUST report the error to the higher level.
3. If the **_status** value is any other error value, the client MUST report the failure to the higher layer.
4. If the **_status** value indicates success, the results MUST be reported to the higher layer requesting the information, and further actions are determined by the higher layer.

For informative purposes, it is expected that higher layers will typically perform the following actions, but these are not enforced by the Windows Search Protocol client:

- If the values in rows represent the document IDs, chapter handles, or bookmark handles, the higher layer will typically store them for use in subsequent operations that involve valid document IDs, chapter handles, or bookmark handles.
- The higher layer will typically store or display or otherwise use the data from row values.
- For the values that were marked as deferred, the higher layer will fetch the value using CPMFetchValueIn messages.
- The seek description is returned back to the higher layer as well, and can be reused or examined by the higher layer.

For informative purposes, if the higher layer requested handles to chapters and bookmarks that were received in the rows, it MAY perform the following:

- Use CPMGetQueryStatusExIn to check on the execution progress of a query as well as additional status information, such as the number of filtered documents, documents remaining to be filtered, the ratio of documents processed by the query, the total number of rows in the query, and the position of the bookmark in the rowset.
- Use CPMGetNotify to request that the server notify the client of rowset changes.
- Use CPMGetApproximatePositionIn to request the approximate position of a bookmark in a chapter.
- Use CPMCompareBmkIn to request a comparison of two bookmarks in a chapter.
- Use CPMRestartPositionIn to request the server changes the location of the cursor to the start of rowset.

3.2.5.3 Receiving a CPMFetchValueOut Response

When the client receives a CPMFetchValueOut message response from the server, the client MUST perform the following:

1. Check whether the **_status** field in the header indicates success or failure. In a case of failure, notify the higher layer. Otherwise, continue as per the following:

2. Check **_fValueExist**, and if set to 0x00000000, notify the higher layer that the value was not found.

3. Otherwise, append **_cbValue** bytes from **vValue** to the Current Property Value.

4. If **_fMoreExists** is set to 0x00000001, increment _Current Bytes Received by **_cbValue** and send a CPMFetchValueIn message to the server, setting **_cbSoFar** to the value of Current Bytes Received, **_cbPropSpec** to zero, and **_cbChunk** to the buffer size required by the higher layer.

5. If **_fMoreExists** is set to 0x00000000, indicate the property value from the Current Property Value to the higher layer.

3.2.5.4 Receiving a CPMFreeCursorOut Response

When the client receives a successful CPMFreeCursorOut message response from the server, the client MUST return the **_cCursorsRemaining** value to the higher layer.

The following information is given for informative purposes only and is not enforced by the Windows Search Protocol client. The higher layer is expected to keep track of cursor handles and not to use those which have already been freed. When the value of **_cCursorsRemaining** is equal to 0x00000000, the higher layer can use the connection to specify another query (using a CPMCreateQueryIn message).

3.2.5.5 Receiving a CPMFindIndicesOut Response

When the client receives a CPMFindIndicesOut message response from the server, the client MUST perform the following:

1. If the **_status** value is an error value, the client MUST report the failure to the higher layer.

2. If the **_status** value indicates success, the results MUST be reported to the higher layer requesting the information, and further actions are determined by the higher layer.

3. If the **_cDepthNext** value is zero, the client MUST report to the higher layer that no occurrence of the given document identifiers was found.

For informative purposes, it is expected that higher layers will typically perform the following actions, but these are not enforced by the Windows Search Protocol client:

- Store the returned coordinate if an occurrence was found and use it to send another CPMFindIndicesIn message in order to retrieve the next occurrence. This procedure can be repeated until no results are found.

3.2.5.6 Receiving a CPMGetRowsetNotifyOut Response

When the client receives a CPMGetRowsetNotifyOut message response from the server, the client MUST perform the following:

1. If the **_status** value is an error value, the client MUST report the failure to the higher layer.

2. If the **_status** value indicates success, the results MUST be reported to the higher layer requesting the information, and further actions are determined by the higher layer.

3. If the **eventType** value indicates PROPAGATE_NONE then no further action is required; otherwise the higher layer is informed of the type of event and relevant information for the event.

3.2.5.7 Receiving a CPMGetScopeStatisticsOut Response

When the client receives a CPMGetScopeStatisticsOut message response from the server, the client MUST perform the following:

1. If the **_status** value is an error value, the client MUST report the failure to the higher layer.
2. If the **_status** value indicates success, the results MUST be reported to the higher layer requesting the information, and further actions are determined by the higher layer.
3. Otherwise the client MUST report the **dwIndexedItems**, **dwOutstandingAdds**, and **dwOutstandingModifies** values returned in the message to the higher layer.

3.2.5.8 Receiving a CPMSetScopePrioritizationOut Response

When the client receives a CPMSetScopePrioritizationOut message response from the server, the client MUST perform the following:

1. If the **_status** value is an error value, the client MUST report the failure to the higher layer.
2. If the **_status** value indicates success, the results MUST be reported to the higher layer requesting the information, and further actions are determined by the higher layer.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Example 1

In the following example, consider a scenario in which the user, UserA on machine UserA-2A, wants to obtain the path files that contain the word "flowers" from the set of files stored on server UserA-4 in catalog SYSTEM. The query, as expressed in the Microsoft Windows Search SQL syntax, is:

[SQL]

```
SELECT Path FROM UserA-4.SystemIndex..Scope() WHERE "SCOPE"
= 'file://UserA-4/Users/UserA/Pictures' AND CONTAINS(*, '"flowers"')
```

Assume that machine UserA-2A is running a 32-bit Windows Vista operating system, and machine UserA-4 is running a 64-bit Windows Server 2008 operating system.

1. The user launches a search application and enters the search query. The application in turn passes the search query to the protocol client.

2. The protocol client establishes a connection with search server UserA-4. The protocol client uses the named pipe \pipe\MsFteWds to connect to the server UserA-4 over SMB.

3. The protocol client then prepares a CPMConnectIn message with the following values:

   - The header of the message is populated as follows:
     - **_msg** is set to 0x000000C8, indicating that this is a CPMConnectIn message.
     - **_status** is set to 0x00000000.
     - **_ulChecksum** contains the checksum, computed as specified in section 3.2.4.
     - **_ulReserved2** is set to 0x00000000.
   - The body of the message is populated as follows:
     - **_iClientVersion** is set to 0x00000109, indicating that the server is to validate the **checksum** field and that the client is running Windows Search 4.0.
     - **_fClientIsRemote** is set to 0x00000001, indicating that the server is a remote server.
     - **_cbBlob1** is set to 340, which is the size, in bytes, of the **cPropSet**, **PropertySet1** and **PropertySet2** fields combined.
     - **_cbBlob2** is set to 1124, which is the size, in bytes of the **cExtPropSet** and **aPropertySet** fields combined.
     - **MachineName** is set to USERA-2A.
     - **UserName** is set to UserA.
     - **cPropSets** is set to 0x00000002.

   Note In the following descriptions of CDbProp structures, several fields always contain default values. They are omitted from the description to improve clarity. The fields with default values are as follows:

   - **DBPROPOPTIONS** is set to 0x0000000.

- **DBPROPSTATUS** is set to 0x00000000.
- For the **colid** element:
  - **eKind** is set to 0x00000001 (DBKIND_GUID_PROPID).
  - GUID is null (all zeros) , meaning the value applies to the query, not to just a single column.
  - **ulID** is set to 0x00000000.

- The **PropertySet1** field is of type CDbPropSet. The CDbPropSet structure comprising the **PropertySet1** field is populated as follows:
  - The **GuidPropSet** field is set to A9BD1526-6A80-11D0-8C9D-0020AF1D740E (DBPROPSET_FSCIFRMWRK_EXT).
  - The **cProperties** field is set to 0x00000004.
  - The **aProps** field is an array of CDbProp structures.
  - For the **aProps[0]** element:
    - PropId is set to 0x00000002 (DBPROP_CI_CATALOG_NAME).
    - For the **vValue** element:
      - **vType** is set to 0x001F (VT_LPWSTR).
      - **vValue** is set to the name of the desired catalog, (for example, "Windows\SYSTEMINDEX").
  - For the **aProps[1]** element:
    - PropId is set to 0x00000007 (DBPROP_CI_QUERY_TYPE).
    - For the **vValue** element:
      - **vType** is set to 0x0003 (VT_I4).
      - **vValue** is set to 0x00000000 (CiNormal), meaning it is a regular query.
  - For the **aProps[2]** element:
    - PropId is set to 0x00000004 (DBPROP_CI_SCOPE_FLAGS).
    - For the **vValue** element:
      - **vType**: 0x1003 (VT_VECTOR | VT_I4).
      - **vValue**: 0x00000001 / 0x00000001 (one element with value 1), meaning search subfolders.
  - For the **aProps[3]** element:
    - PropId is set to 0x00000003 (DBPROP_CI_INCLUDE_SCOPES).
    - For the **vValue** element:
      - **vType** is set to 0x101F (VT_VECTOR | VT_LPWSTR).
      - **vValue** is set to 0x00000001 / 0x00000002 / "\" (one element with a two-character null-terminated string), meaning the root scope.

- The **PropertySet2** field is of type CDbPropSet. The CDbPropSet structure comprising the **PropertySet1** field is populated as follows:
  - **GuidPropSet** is set to AFAFACA5-B5D1-11D0-8C62-00C04FC2DB8D (DBPROPSET_CIFRMWRKCORE_EXT).
  - The **cProperties** field is set to 0x00000001.
  - The **aProps** field is an array of CDbProp structures.
  - For the **aProps[0]** element:
    - **PropId** is set to 0x00000002 (DBPROP_MACHINE).
    - For **vValue** element:
      - **vType**: 0x0008 (VT_BSTR).
      - **vValue**: 0x10 / "USERA-4" (16 bytes / null-terminated Unicode string), meaning "USERA-4", the name of a server.
- The **cExtPropSet** field is set to 0x00000004.
- **aPropertySets[0]** is of type CDbPropSet and contains the following fields:
  - The **GuidPropSet** field is set to GUID: AA6EE6B0-E828-11D0-B23E-00AA0047FC01 (DBPROPSET_MSIDXS_ROWSETEXT).
  - The **cProperties** field is set to 0x00000006.
  - The **aProps** field is an array of CDbProp structures.
  - For the **aProps[0]** element:
    - **PropId** is set to 0x00000002 (MSIDXSPROP_ROWSETQUERYSTATUS).
    - For the **vValue** element:
      - **vType** is set to 0x0003 (VT_I4).
      - **vValue** is set to 0x0000. This value is ignored by all Windows implementations of the protocol.
  - For the **aProps[1]** element:
    - **PropId** is set to 0x00000003 (MSIDXSPROP_COMMAND_LOCALE_STRING).
    - For the **vValue** element:
      - **vType** is set to 0x0008 (VT_BSTR).
      - **vValue** is set to 0x6 / "EN" (6 bytes / null-terminated Unicode string), meaning the user locale is English.
  - For the **aProps[2]** element:
    - **PropId** is set to 0x00000004 (MSIDXSPROP_QUERY_RESTRICTION).
    - For the **vValue** element:
      - **vType** is set to 0x0008 (VT_BSTR).

- **vValue** is set to 0x2 / "" (2 bytes / null-terminated empty Unicode string). This value is ignored by all Windows implementations of the protocol.

- For the **aProps[3]** element:
  - PropId is set to 0x00000005 (MSIDXSPROP_PARSE_TREE).
  - For the **vValue** element:
    - **vType** is set to 0x0008 (VT_BSTR).
    - **vValue** is set to 0x2 / "" (2 bytes / null-terminated empty Unicode string). This value is ignored by all Windows implementations of the protocol.
- For the **aProps[4]** element:
  - PropId is set to 0x00000006 (MSIDXSPROP_MAX_RANK).
  - For the **vValue** element:
    - **vType** is set to 0x0003 (VT_I4).
    - **vValue** is set to 0x0000. This value is ignored by all Windows implementations of the protocol.
- For the **aProps[5]** element:
  - PropId is set to 0x00000007 (MSIDXSPROP_RESULTS_FOUND).
  - For the **vValue** element:
    - **vType** is set to 0x0003 (VT_I4).
    - **vValue** is set to 0x0000. This value is ignored by all Windows implementations of the protocol.

- aPropertySets[1] is of type CDbPropSet and contains the following fields:
  - The **GuidPropSet** field is set to GUID: A7AC77ED-F8D7-11CE-A798-0020F8008025 (DBPROPSET_QUERYEXT).
  - The **cProperties** field is set to 0x0000000A.
  - The **aProps** field is an array of CDbProp structures.
  - For the **aProps[0]** element
    - **PropId** is set to 0x00000002 (DBPROP_USECONTENTINDEX).
    - For the **vValue** element:
      - **vType** is set to 0x000B (VT_BOOL).
      - **vValue** is set to 0x0000 (FALSE), meaning forced use of the full text index is FALSE.
  - For the **aProps[1]** element:
    - PropId is set to 0x00000003 (DBPROP_DEFERNONINDEXEDTRIMMING).
    - For the **vValue** element:
      - **vType** is set to 0x000B (VT_BOOL).

        - **vValue** is set to 0x0000 (FALSE), meaning trimming of security results will not be deferred.
    - For the **aProps[2]** element:
        - PropId is set to 0x00000004 (DBPROP_USEEXTENDEDDBTYPES).
        - For the **vValue** element:
            - **vType** is set to 0x000B (VT_BOOL).
            - **vValue** is set to 0x0000 (FALSE), meaning extended DB types are not used.
    - For the **aProps[3]** element:
        - PropId is set to 0x00000005 (DBPROP_IGNORENOISEONLYCLAUSES).
        - For the **vValue** element:
            - **vType** is set to 0x000B (VT_BOOL).
            - **vValue** is set to 0x00 (FALSE), meaning full text clauses consisting entirely of noise words will result in an error being returned.
    - For the **aProps[4]** element:
        - PropId is set to 0x00000006 (DBPROP_GENERICOPTIONS_STRING).
        - For the **vValue** element:
            - **vType** is set to 0x0008 (VT_BSTR).
            - **vValue** is set to 0x2 / "" (2 bytes / null-terminated empty Unicode string), meaning no generic options were set.
    - For the **aProps[5]** element:
        - PropId is set to 0x00000008 (DBPROP_DEFERCATALOGVERIFICATION).
        - For the **vValue** element:
            - **vType** is set to 0x000B (VT_BOOL).
            - **vValue** is set to 0x0000 (FALSE), meaning catalog verification is not deferred.
    - For the **aProps[6]** element:
        - PropId is set to 0x0000000E (DBPROP_IGNORESBRI).
        - For the **vValue** element:
            - **vType** is set to 0x000B (VT_BOOL).
            - **vValue** is set to 0x0000 (FALSE), meaning the query can use the sort-by-rank index optimization.
    - For the **aProps[7]** element:
        - **PropId** is set to 0x0000000A (DBPROP_GENERATEPARSETREE).
        - For the **vValue** element:
            - **vType** is set to 0x000B (VT_BOOL).

- **vValue** is set to 0x0000 (FALSE), meaning a parse tree is not generated for debugging.

- For the **aProps[8]** element:
  - **PropId** is set to 0x0000000C (DBPROP_FREETEXTANYTERM).
  - For the **vValue** element:
    - **vType** is set to 0x000B (VT_BOOL).
    - **vValue** is set to 0x0000 (FALSE), meaning all terms from a FREETEXT clause appear in every matching document.
- For the **aProps[9]** element:
  - **PropId** is set to 0x0000000D (DBPROP_FREETEXTUSESTEMMING).
  - For the **vValue** element:
    - **vType** is set to 0x000B (VT_BOOL).
    - **vValue** is set to 0x0000 (FALSE), meaning stemming is not used when interpreting a FREETEXT clause.

- aPropertySets[2] is of type CDbPropSet and contains the following fields:
  - The **GuidPropSet** field is set to GUID: B5D1AFAF-11D0-628C-00C0-4FC2DB8D0000 (DBPROPSET_CIFRMWRKCORE_EXT).
  - The **cProperties** field is set to 0x00000001.
  - The **aProps** field is an array of CDbProp structures.
  - For the **aProps[0]** element:
    - PropId is set to 0x00000002 (DBPROP_MACHINE).
    - For the **vValue** element:
      - **vType** is set to 0x0008 (VT_BSTR).
      - **vValue** is set to 0x0010 / "USERA-4" (7 Unicode characters + null = 0x10 bytes), meaning the target server is named USERA-4.
- **aPropertySets[3]** is of type CDbPropSet and contains the following fields:
  - The **GuidPropSet** field is set to GUID: A9BD1526-6A80-11D0-8C9D-0020AF1D740E (DBPROPSET_FSCIFRMWRK_EXT)
  - The **cProperties** field is set to 0x00000003.
  - The **aProps** field is an array of CDbProp structures.
  - For the **aProps[0]** element:
    - **PropId** is set to 0x00000003 (DBPROP_CI_INCLUDE_SCOPES).
    - For the **vValue** element:
      - **vType** is set to 0x2008 (VT_ARRAY | VT_BSTR).
      - **vValue** is set to:

- cDims = 0x0001
- fFeatures = <ignore 2 bytes>
- cbElements = 0x00000004 (4 bytes per element)
- SAFEARRAYBOUND[0].cElements = 0x00000001 (1 element)
- SAFEARRAYBOUND[0].lLbound = 0x00000000 (0-based)
- element = 0x00000004 / "\" (1 Unicode character plus null), meaning search everywhere ("\" is the root scope)

- For the **aProps[1]** element:
  - PropId is set to 0x00000004 (DBPROP_CI_SCOPE_FLAGS).
  - For the **vValue** element:
    - **vType** is set to 0x2003 (VT_ARRAY | VT_I4).
    - **vValue** is set to:
      - cDims = 0x0001
      - fFeatures = <ignore 2 bytes>
      - cbElements = 0x00000004 (4 bytes per element)
      - SAFEARRAYBOUND[0].cElements = 0x00000001 (1 element)
      - SAFEARRAYBOUND[0].lLbound = 0x00000000 (0-based)
      - element = 0x00000001, meaning QUERY_DEEP.
- For the **aProps[2]** element:
  - PropId is set to 0x00000002 (DBPROP_CI_CATALOG_NAME).
  - For the **vValue** element:
    - **vType** is set to 0x0008 (VT_BSTR).
    - **vValue** is set to 0x00000028 / and the name of the catalog to use, for example, "Windows\SYSTEMINDEX" (19 Unicode characters plus null), meaning the catalog to use is named Windows\SYSTEMINDEX.
- Various padding fields are filled in as needed. The message is sent to the server.

4. The server verifies that the **_ulChecksum** is correct, verifies that the user is authorized to make this request, and responds with a CPMConnectOut message.
   - The header of the message is populated as follows:
     - **_msg** is set to 0x000000C8, indicating that this is a CPMConnectOut message.
     - **_status** is set to 0x0000000, indicating SUCCESS.
     - **_ulChecksum** is set to 0.
     - **_ulReserved2** is set to 0x00000000.
   - The body of the message is populated as 0x00010109 (64-bit Windows Vista).

5. The **_reserved** fields are filled with arbitrary data.
6. The client prepares a CPMCreateQueryIn message.
    - The header of the message is populated as follows:
        - **_msg** is set to 0x000000CA, indicating that this is a CPMCreateQueryIn message.
        - **_status** is set to 0x00000000.
        - **_ulChecksum** contains the checksum, computed according to section 3.2.5.
        - **_ulReserved2** is set to 0x00000000.
    - The body of the message is populated as follows:
        - The **Size** field is set to the size of the rest of the message.
        - The **CColumnSetPresent** field is set to 0x01.
    - The **ColumnSet** field is of type CColumnSet. The CColumnSet structure comprising this field is set as follows:
        - The **_count** field is set to 0x00000001, indicating one column is returned.
        - The indexes array is 0x00000000, indicating the first entry in **_aPropSpec**.
    - The **CRestrictionPresent** field is set to 0x01, indicating the **RestrictionArray** field is present.
        - Set **RestrictionArray** such that:
            - **Count** is set to 0x01.
            - **isPresent** is set to 0x01 (restriction present).
        - The **Restriction** field is of type CRestriction and is set to:
            - **_ulType** is set to 0x00000001 (RTAnd).
            - **_weight** is set to 0x000003E8.
        - The rest of the field contains a CNodeRestriction structure:
            - **_cNode** is set to 0x00000002, meaning two clauses are AND-ed together.
            - **_paNode[0]** is of type CRestriction and is set to:
                - **_ulType** is set to 0x00000005 (RTProperty.
                - **_weight** is set to 0x000003E8.
                - The rest of the field contains a CPropertyRestriction structure:
                    - **_relop** is set to 0x00000004 (PREQ), meaning a property is tested for equality.
                    - **_Property** is a CFullPropspec structure that is set to GUID B725F130-47EF-101A-A5-F1-02608C9EEBAC / 0x00000001 (for PRSPEC_PROPID) /0x00000016, which represents the Windows scope property.
                    - **_PrVal** is a CBaseStorageVariant:

- **vType** is set to 0x001F (VT_LPWSTR).
- **vValue** is set to the Unicode string "file://UserA-4/Users/UserA/Pictures", which is the scope to match.

- **_lcid** is 0x0409, meaning U.S. English.

- **_paNode[1]** contains:
  - **_ulType** is set to 0x00000004 (RTContent.
  - **_weight** is set to 0x000003E8.
  - The rest of the field contains a CContentRestriction structure:
    - **_Property** is set to GUID 49691C90-7E17-101A-A91C-08002B2ECDA9/ 0x00000001 (for PRSPEC_PROPID) / 0x00000006, meaning DISPID_QUERY_ALL (for example, search across all properties).
    - **_Cc** is set to 0x00000007.
    - **_pwcsphrase** is set to the string "flowers".
    - **_lcid** is set to 0x409 (for English).
    - **_ulGenerateMethod** is set to 0x00000000 (exact match).

- **CSortPresent** is set to 0x00.
- **CCategorizationSetPresent** is set to 0x00.

- **RowSetProperties** is set as follows:
  - **_uBooleanOptions** is set to 0x00000001 (sequential).
  - **_ulMaxOpenRows** is set to 0x00000000.
  - **_ulMemoryUsage** is set to 0x00000000.
  - **_cMaxResults** is set to 0x00000000 (return all rows).
  - **_cCmdTimeOut** is set to 0x0000001E (timeout after 30ms).

- **PidMapper** is set to:
  - **_count** is set to 0x00000003.
  - **_aPropSpec[0]** is set to GUID B725F130-47EF-101A-A5-F1-02608C9EEBAC / 0x00000001 (for PRSPEC_PROPID)/0x0000000B, which represents the Windows path property.
  - **_aPropSpec[1]** is set to GUID B725F130-47EF-101A-A5-F1-02608C9EEBAC / 0x00000001 (for PRSPEC_PROPID)/0x00000016, which represents the Windows scope property.
  - **_aPropSpec[2]** is set to GUID 49691C90-7E17-101A-A91C-08002B2ECDA9/ 0x00000001 (for PRSPEC_PROPID)/0x00000006, which represents the "all properties" property.
  - **ColumnGroupArray** is set to 0x00000000, meaning no weights for columns are specified.
  - **_lcid** is set to 0x00000409, meaning U.S. English is the default locale for the query.

7. The server processes it and responds with a CPMCreateQueryOut message.
    - The header of the message is populated as follows:
        - **_msg** is set to 0x000000CA, indicating that this is a CPMCreateQueryOut message.
        - **_status** is set to SUCCESS.
        - **_ulChecksum** is set to 0x00000000 (or any other arbitrary value).
        - **_ulReserved2** is set to 0x00000000 (or any other arbitrary value).
    - The body of the message is populated as follows:
        - **_fTrueSequential** is set to 0x00000001, indicating that the query cannot use an index.
        - **_fWorkIdUnique** is set to 0x00000001.
        - The **aCursors** array contains only one element, representing a cursor handle to this query. The value depends on the state of the server, assuming that the returned value is 0xAAAAAAAA.
8. The client issues a CPMSetBindingsIn request message to define the format of a row.
    - The header of the message is populated as follows:
        - **_msg** is set to 0x000000D0, indicating that this is a CPMSetBindingsIn message.
        - **_status** is set to SUCCESS.
        - **_ulChecksum** is set to 0x00000000 (or any other arbitrary value).
        - **_ulReserved2** is set to 0x00000000 (or any other arbitrary value).
    - The body of the message is populated as follows:
        - **_hCursor** is set to 0xAAAAAAAA.
        - **_cbRow** is set to 0x20 (big enough to fit columns).
        - **_cbBindingDesc** is set to 0x61, the size of the **_cColumns** and **_aColumns** fields combined.
        - **_dummy** is set to an arbitrary value such as 0x0A00000C.
        - **_cColumns** is set to 0x00000002.
        - The **_aColumns** array is set to contain two CTableColumn structures.
        - **_aColumns[0]** contains the following:
            - **_PropSpec** is set to GUID B725F130-47EF-101A-A5-F1-02608C9EEBAC / 0x00000001 (for PRSPEC_PROPID) / 0x0000000B, which represents the Windows path property.
            - **vType** is set to 0x000C (VT_VARIANT).
            - **_AggregateStored** is set to 0x01, meaning aggregate info is present.
            - **_AggregateType** is set to 0x00, meaning the query is not aggregated.
            - **_ValueUsed** is set to 0x01 (column transferred in row).

- **_ValueOffset** is set to 0x0008 (8 bytes from beginning of row).
- **_ValueSize** is set to 0x0010 (size of a VT_VARIANT).
- **_StatusUsed** is set to 0x01 (status is reported in row).
- **_StatusOffset** is set to 0x0002.
- **_LengthUsed** is set to 0x01 (length is reported).
- **_LengthOffset** is set to 0x0004.

- **_aColumns[1]** contains the following:
  - **_PropSpec** is set to GUID 49691C90-7E17-101A-A91C-08002B2ECDA9/ 0x00000001 (for PRSPEC_PROPID)/0x00000005, which represents the private internal document ID of an indexed document.
  - **_vType** is set to 0x0003 (VT_I4).
  - **_AggregateStored** is set to 0x01, meaning aggregate info is present.
  - **_AggregateType** is set to 0x00, meaning the query is not aggregated.
  - **_ValueUsed** is set to 0x01 (column transferred in row).
  - **_ValueOffset** is set to 0x0018 (24 bytes from beginning of row).
  - **_ValueSize** is set to 0x0004 (size of a VT_I4).
  - **_StatusUsed** is set to 0x01 (status is reported in row).
  - **_StatusOffset** is set to 0x0003.
  - **_LengthUsed** is set to 0x00 (length is not reported).

9. The server processes it and responds with the header of a CPMSetBindingsIn message.
   - The header of the message is populated as follows:
     - **_msg** is set to 0x000000D0.
     - **_status** is set to SUCCESS.
     - **_ulChecksum** is set to 0x00000000 (or any other arbitrary value).
     - **_ulReserved2** is set to 0x00000000 (or any other arbitrary value).
10. The client issues a CPMGetRowsIn request message, assuming that the client is prepared to accept 32 rows at this point, and requests them in ascending order.
    - The header of the message is populated as follows:
      - **_msg** is set to 0x000000CC, indicating that this is a CPMGetRowsIn message.
      - **_status** is set to 0x00000000.
      - **_ulChecksum** contains the checksum, computed according to section 3.2.4.
      - **_ulReserved2** is set to 0x00000000.
    - The body of the message is populated as follows:

- **_hCursor** is set to 0xAAAAAAAA.
- **_cRowsToTransfer** is set to 0x00000014.
- **_cRowWidth** is set to 0x00000020 (from bindings).
- **_cbSeek** is set to 0x0000000C, which is the size of the **eType**, **_chapt**, and **CRowSeekNext** fields combined.
- **_cbReserved** is set to 0x0x20 (0x14 plus **_cbSeek**).
- **_cbReadBuffer** is set to 0x4000 (because 1000 * 0x20 > 0x4000).
- **_ulClientBase** is set to 0x03C924C8 (execution dependent).
- **_fBwdfetch** is set to 0x00000000, indicating that the rows are to be fetched in forward order.
- **eType** is set to 0x0000001, indicating that the client wants next rows.
- **_chapt** is set to 0 (not a chaptered result).
- **eType** is set to 0x00000000 (eRowsSeekNone), meaning start with next 0x00000000.

- The body of the message is populated as follows:
  - **_CRowsReturned** is set to 0x00000002.
  - **eType** is set to 0x00000000 (eRowsSeekNone).
  - **_chapt** is set to 0x00000000 (not a chaptered result).
  - Rows contain the size of the two documents that contain the word "flowers". The raw buffer will look similar to the following (-- indicates unused space):

```
-- -- 00 00 (status OK for both columns)
7E 00 00 00 (length = 0x7E - 0x10 (inline) = 0x6E bytes of string)
1F 00 -- -- (VT_LPWSTR)
-- -- -- --
58 64 C9 03 (address of VT_LPWSTR: 0x03C96458 - base 0x03C924C8 =
             offset 0x3F90 into buffer)
-- -- -- --
96 02 00 00 (WorkId = 0x296)
-- -- -- --

-- -- 00 00 (status OK for both columns)
86 00 00 00 (length = 0x86 - 0x10 (inline) = 0x76 bytes of string)
1F 00 -- -- (VT_LPWSTR)
-- -- -- --
E0 63 C9 03 (address of VT_LPWSTR: 0x03C963E0 - base 0x03C924C8 =
             offset 0x3F18 into buffer)
-- -- -- --
97 02 00 00 (WorkId = 0x297)
-- -- -- --
```

  - And at offset 0x3F90 into the buffer will be a Unicode string of length 55 (including null) "file://UserA-4/Users/UserA/Pictures/forest flowers.jpg".
  - And at offset 0x3F18 into the buffer will be a Unicode string of length 59 (including null) "file://UserA-4/Users/UserA/Pictures/frangipani flowers.jpg".

11. The client issues another CPMGetRowsIn request message to look for more rows.

- The header of the message is populated as follows:
  - **_msg** is set to 0x000000CC, indicating that this is a CPMGetRowsIn message.
  - **_status** is set to 0x00000000.
  - **_ulChecksum** contains the checksum, computed according to section 3.2.4.
  - **_ulReserved2** is set to 0x00000000.
- The body of the message is populated as follows:
  - **_hCursor** is set to 0xAAAAAAAA.
  - **_cRowsToTransfer** is set to 0x00000014.
  - **_cRowWidth** is set to 0x00000020 (from bindings).
  - **_cbSeek** is set to 0x0000000C, which is the size of the **eType**, **_chapt**, and **CRowSeekNext** fields combined.
  - **_cbReserved** is set to 0x0x20 (0x14 plus _cbSeek).
  - **_cbReadBuffer** is set to 0x4000 (because 1000 * 0x20 > 0x4000).
  - **_ulClientBase** is set to 0x03C924C8 (execution dependent).
  - **_fBwdfetch** is set to 0x00000000, indicating that the rows are to be fetched in forward order.
  - **eType** is set to 0x0000001, indicating that the client wants next rows.
  - **_chapt** is set to 0 (not a chaptered result).
  - **eType** is set to 0x00000000 (eRowsSeekNone), I meaning start with next available row.

12. The server processes it and responds with a CPMGetRowsOut message, assuming the server found no more documents that contain the word "flowers".
    - The header of the message is populated as follows:
      - **_msg** is set to 0x000000CC, indicating that this is a CPMGetRowsOut message.
      - **_status** is set to SUCCESS.
      - **_ulChecksum** is set to 0x00000000.
      - **_ulReserved2** is set to 0x00000000.
    - The body of the message is populated as follows:
      - **_CRowsReturned** is set to 0x00000000, meaning no more rows are available.
13. The client sends a CPMFreeCursorIn message to close the handle to the query.
    - The header of the message is populated as follows:
      - **_msg** is set to 0x000000CB, indicating that this is a CPMFreeCursorIn message.
      - **_status** is set to 0x00000000.
      - **_ulChecksum** contains the checksum, computed according to section 3.2.4.

        - **_ulReserved2** is set to 0x00000000.
    - The body of the message is populated as follows:
        - **_hCursor** is set to 0xAAAAAAAA.

14. The server processes it and responds with a CPMFreeCursorOut message.
    - The header of the message is populated as follows:
        - **_msg** is set to 0x000000CB, indicating that this is a CPMFreeCursorOut message.
        - **_status** is set to 0x00000000.
        - **_ulChecksum** contains the checksum, computed according to section 3.2.4.
        - **_ulReserved2** is set to 0x00000000.
    - The body of the message is populated as follows:
        - **_cCursorsRemaining** is set to 0x00000000, meaning no more cursors are active for the query.
15. The client sends a CPMDisconnect message to end the connection.
    - The header of the message is populated as follows:
        - **_msg** is set to 0x000000C9, indicating that this is a CPMDisconnect message.
        - **_status** is set to 0x00000000.
        - **_ulChecksum** is set to 0x00000000.

The server processes the message and removes all client states.

5 Security

The following sections specify security considerations for administrators.

5.1 Security Considerations for Implementers

For indexing implementations that index secure content, consider using the user context provided by [MS-SMB] to trim search results and return only those results accessible to the caller.

5.2 Index of Security Parameters

The only security parameter is impersonation level, section 2.1.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include ~~released service packs~~updates to those products.

Windows Search 4.0 is an out-of-band release that can be installed as an update to (exclusively) Windows XP operating system, Windows Vista operating system, Windows Server 2008 operating system, Windows Home Server server software, and all versions of Windows Server 2003 operating system. Windows 7 operating system and Windows Server 2008 R2 operating system cannot have Windows Search 4.0 installed. Windows Search 4.0 is the only out-of-band release of Windows Search. All other versions of Windows Search came with the operating system and can be identified as such.

With regard to Windows Search behavior, Windows Vista (without Windows Search 4.0) and Windows Server 2008 (without Windows Search 4.0) are equivalent. Windows XP, Windows Server 2003, Windows Server 2003 R2 operating system, Windows Vista, Windows Server 2008, and Windows Home Server, all with Windows Search 4.0, are equivalent. Windows 7 and Windows Server 2008 R2 are equivalent. The equivalent versions can be used interchangeably.

- Windows XP
- Windows Server 2003
- Windows Server 2003 R2
- Windows Vista
- Windows Server 2008
- Windows 7
- Windows Server 2008 R2
- Windows Home Server
- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system
- Windows 10 operating system
- Windows Server 2016 operating system
- Windows Server operating system

Exceptions, if any, are noted ~~below.~~in this section. If ~~a~~ an update version, service pack or ~~Quick Fix Engineering (QFE~~Knowledge Base (KB) number appears with ~~the~~a product ~~version,~~name, the behavior changed in that ~~service pack or QFE.~~update. The new behavior also applies to subsequent ~~service packs of the product~~updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

<1> Section 1.8: Windows uses only the values specified in [MS-ERREF].

<2> Section 2.1: Applications typically interact with an OLE DB interface wrapper (as specified in [MSDN-OLEDBP]), for example, a protocol client, and not directly with the protocol.

<3> Section 2.2.1.13: In Windows Vista, the default catalog name is SystemIndex.

<4> Section 2.2.3.1: This value is usually zero, except immediately after indexing has been started or after a notification queue overflows.

<5> Section 2.2.3.2: On Windows-based clients, the **iClientVersion** is set as follows.

Value	Meaning
0x00000102	Client OS is either 32-bit Windows Server 2008, or 32-bit Windows Vista.
0x00000109	Client OS is either 32-bit Windows XP, 32-bit Windows Server 2003, 32-bit Windows Vista with Windows Search 4.0, 32-bit Windows Server 2003 with Windows Search 4.0. All of these versions of Windows are running Windows Search 4.0.
0x000010102	64-bit version of Windows Vista or Windows Server 2008.
0x00010109	64-bit version of Windows Vista or Windows Server 2008 with Windows Search 4.0 installed.

<6> Section 2.2.3.2: On Windows 7 and Windows Server 2008 R2 operating system, the values are as follows.

Value	Meaning
0x00000700	32-bit Windows 7.
0x00010700	64-bit Windows 7 or Windows Server 2008 R2.

<7> Section 2.2.3.3: On Windows-based clients, the **_serverVersion** is set as follows.

Value	Meaning
0x00000102	OS is either 32-bit Windows Server 2008, 32-bit Windows Home Server, or 32-bit Windows Vista – all without Windows Search 4.0 installed.
0x00000109	OS is either 32-bit Windows XP, 32-bit Windows Server 2003, 32-bit Windows Vista with Windows Search 4.0, 32-bit Windows Server 2003 – all with Windows Search 4.0.
0x00010102	64-bit version of Windows Vista64-bit Windows Home Server, or Windows Server 2008 – all without Windows Search 4.0 installed.
0x00010109	64-bit version of Windows Vista or Windows Server 2008 – all with Windows Search 4.0 installed.

<8> Section 2.2.3.3: On Windows 7 and Windows Server 2008 R2, the values are as follows.

Value	Meaning
0x00000700	32-bit Windows 7.
0x00010700	64-bit Windows 7 and Windows Server 2008 R2.

<9> Section 2.2.3.3: Introduced with Windows Vista operating system with Service Pack 2 (SP2) and included in Windows 7 and Windows Server 2008 R2.

<10> Section 2.2.3.3: Introduced with Windows Vista SP2 and included in Windows 7 and Windows Server 2008 R2.

<11> Section 2.2.3.3: Introduced with Windows Vista SP2 and included in Windows 7 and Windows Server 2008 R2.

<12> Section 2.2.3.3: Introduced with Windows Vista SP2 and included in Windows 7 and Windows Server 2008 R2.

<13> Section 2.2.3.14: Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 can return zero for this field depending, in part, on when the CPMRatioFinishedIn and CPMRatioFinishedOut messages are exchanged. When zero is returned, the client is assumed to ignore the information, as the correct information is not yet available. Note that Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008 require separate installation of Windows Search 4.0 in order for this to be allowed.

<14> Section 2.2.3.15: This field (buffer size) is set in Windows to 0x00004000.

<15> Section 2.2.3.18: Windows Search 4.0 (WS 4.0) does not support CPMSendNotifyOut and does not inform the client. Do not use values of **_watchNotify** under WS 4.0.

<16> Section 2.2.3.19: Not implemented on Windows Vista. Not implemented on Windows XP with Windows Desktop Search 3.0. Implemented on Windows 7 and Windows Search 4.0.

<17> Section 2.2.3.21: Not implemented on Windows Vista. Not implemented on Windows XP with Windows Desktop Search 3.0. Implemented on Windows 7 and Windows Search 4.0.

<18> Section 2.2.3.23: Not implemented on Windows Vista. Not implemented on Windows XP with Windows Desktop Search 3.0. Implemented on Windows 7 and Windows Search 4.0.

<19> Section 2.2.3.27: This message was added in Windows 7 and Windows Server 2008 R2.

<20> Section 2.2.3.29: This message was added in Windows 7 and Windows Server 2008 R2.

<21> Section 2.2.3.31: This message was added in Windows 7 and Windows Server 2008 R2.

<22> Section 2.2.3.33: This message was added in Windows 7 and Windows Server 2008 R2.

<23> Section 2.2.4: The same pipe connection is used for future messages, except when the error is returned in a CPMConnectOut message. In the latter case, the pipe connection is terminated.

<24> Section 3: Windows Server 2003 can be used for client and server if Windows® Search Version 4 is installed. It can serve as a client if Windows® Desktop Search Version 3 is installed.

Windows XP can be used for client and server if Windows® Search Version 4 is installed. It can serve as a client if Windows® Desktop Search Version 3 is installed.

Windows Home Server uses MS-WSP as a server and ships with Windows® Desktop Search Version 3.

<25> Section 3.1.5.2.1: In versions before Windows 7 and Windows Server 2008 R2, the search service does not check for the existence of the catalog.

<26> Section 3.1.5.2.3: Cursor handles are not checked before Windows 7 and Windows Server 2008 R2. Before Windows 7 and Windows Server 2008 R2, invalid handles stop the search service. Otherwise, the server will return *E_FAIL* if the *ContainsHandle* output parameter is not true.

<27> Section 3.1.5.2.4: Cursor handles are not checked before Windows 7 and Windows Server 2008 R2. Before Windows 7 and Windows Server 2008 R2, invalid handles stop the search service. Otherwise, the server will return E_FAIL if the *ContainsHandle* output parameter is not true.

<28> Section 3.1.5.2.5: Cursor handles are not checked before Windows 7 and Windows Server 2008 R2. Before Windows 7 and Windows Server 2008 R2, invalid handles stop the search service. Otherwise, the server will return E_FAIL if the *ContainsHandle* output parameter is not true.

<29> Section 3.1.5.2.6: Cursor handles are not checked before Windows 7 and Windows Server 2008 R2. Before Windows 7 and Windows Server 2008 R2, invalid handles stop the search service. Otherwise, the server will return E_FAIL if the *ContainsHandle* output parameter is not true.

<30> Section 3.1.5.2.6: If the CPMSetBindingsIn call fails with a 32-bit client and a 64-bit server, the error returned MUST be STATUS_INVALID_PARAMETER (0xC000000D) rather than E_UNEXPECTED (0x8000FFFF).

<31> Section 3.1.5.2.8: Cursor handles are not checked before Windows 7 and Windows Server 2008 R2. Before Windows 7 and Windows Server 2008 R2, invalid handles stop the search service. Otherwise, the server will return E_FAIL if the *ContainsHandle* output parameter is not true.

<32> Section 3.1.5.2.8: Row width is not checked between 32-bit and 64-bit systems in Windows.

<33> Section 3.1.5.2.9: If the server version is Windows 7 or Windows Server 2008 R2 and the client is running on a WSS version: Windows XP or Windows Server 2003 or Windows Server 2003 R2 or Windows Vista or Windows Server 2008 operating system, this functionality is not implemented and the server MUST report an E_NOTIMPL error.

<34> Section 3.1.5.2.13: Windows Vista, Windows Search 4.0, and Windows Server 2008 return ERROR_INVALID_PARAMETER (0x80070057). Windows 7 and Windows Server 2008 R2 return STATUS_INVALID_PARAMETER (0xC000000D).

<35> Section 3.1.7: This is set to false on any machine running Windows 7 and on any machine running Windows Search 4.0. Otherwise, the value is as described in section 2.2.3.5.

<36> Section 3.1.7: STAT_DONE implies that the server is ready to return rows to the client. This is always set to STAT_DONE on any machine running Windows 7 and on any machine running Windows Search 4.0, because the server is always ready to return rows.

On any previous WSS version, STAT_DONE is not returned until the server is completely done processing the query. If the rows are not ready, it will return STAT_BUSY. Once it is returned, any future call to the **GetRows** abstract interface with the same **QueryIdentifier** argument will successfully return results, if any are left.

<37> Section 3.1.7: This interface is available only on Windows 7.

<38> Section 3.1.7: This is only implemented on Windows Vista, not on Windows Search 4.0 or Windows 7.

<39> Section 3.1.7: This interface is only available on Windows 7.

<40> Section 3.1.7: This interface is only available on Windows 7.

<41> Section 3.1.7: This interface is only available in Windows 7.

<42> Section 3.1.7: This interface is only available in Windows 7.

<43> Section 3.2.4.2.4: For a 32-bit client talking to a 32-bit server, or a 64-bit client talking to a 64-bit server, this value is set to a memory address of the receiving buffer in the application process. This allows for pointers received in the **Rows** field of CPMGetRowsOut to be correct memory pointers in a client application process. Otherwise, it is set to 0x00000000.

7 Change Tracking

~~No table of~~ This section identifies changes ~~is available. The~~ that were made to this document ~~is either new or has had no changes~~ since ~~its~~the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
6 Appendix A: Product Behavior	Added Windows Server to the list of applicable products.	Major

8 Index

A

C

D

N

O

P

Q

R

S

T

V